MAKE | BUILD | HACK | CREATE

HackSpace

TECHNOLOGY IN YOUR HANDS

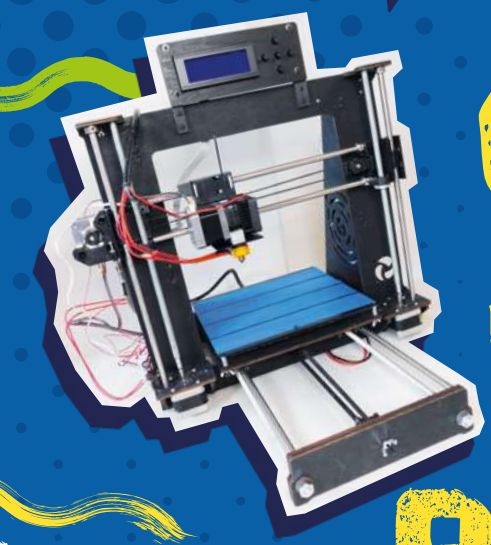
hsmag.CC | April 2021 | Issue #41

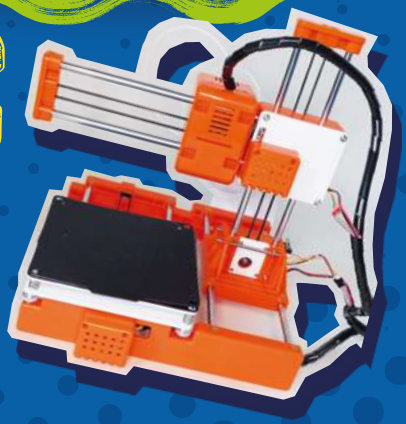
PICO Graphics

Add pixels to your Raspberry Pi microcontroller

BUDGET 3D PRINTING

Get prints this good for under £100





WHAT TO LOOK FOR!

WHAT TO BUY!

WHAT TO AVOID!

Geeky Faye

Fine art meets digital creativity

Woodcut Printing

Use a laser cutter for inky designs

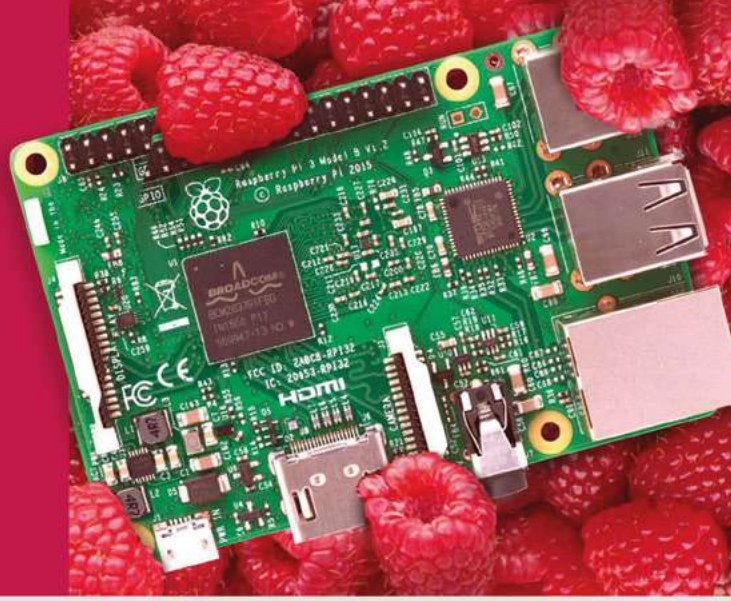
9 772515 514006 41

Raspberry Pi PRESS

Apr. 2021
Issue #41 £6

GLASS-BLOWING ZIPS HIFIVE LOGIC

American **Raspberry Pi** Shop



- Displays
- HATs
- Sensors
- Cases
- Arcade
- Swag
- Project Kits
- Cameras
- Power Options
- Add-on Boards
- Cables and Connectors
- GPIO and Prototyping

Partner and official reseller for top Pi brands:

adafruit · sparkfun ELECTRONICS · PIMORONI · HiFiBerry · and many others!

Price, service, design, and logistics support for **VOLUME PROJECTS**

USA PiShop.us

Canada BuyaPi.ca

Raspberry Pi APPROVED RESELLER

Welcome to HackSpace magazine

What is a budget 3D printer? It's no secret that you can get a good 3D printer for £300 these days. If you shop smart, you can pick up a good printer for £200, but what if you go lower? Our 3D printer test lab has been a hive of activity this month as we aimed to answer the question of 'how good a printer can you get for under £100'?

We've tried printers in this price bracket before and not had good results, but time and technology move on and there are some more options now. The results certainly surprised us. Take a look at page 38 for the full rundown.

If you shop smart, you can pick up a good printer for £200, **but what if you go lower?**

Of course, it's not all 3D printing. On page 74 you'll find a guide to expanding the graphical capabilities of Raspberry Pi Pico and, on page 110, we take a look at adding buttons and lights to this microcontroller. If you're in a less digital frame of mind, you can read all about making things with wood and ink on page 80. Happy making!

BEN EVERARD

Editor ben.everard@raspberrypi.com

Got a comment, question, or thought about HackSpace magazine?

get in touch at **hsmag.cc/hello**

GET IN TOUCH

hackspace@raspberrypi.com

hackspacemag

hackspacemag

ONLINE

hsmag.cc

Available on the App Store

GET IT ON Google Play

EDITORIAL

Editor
Ben Everard
ben.everard@raspberrypi.com

Features Editor
Andrew Gregory
andrew.gregory@raspberrypi.com

Sub-Editors
David Higgs, Nicola King

DESIGN

Critical Media
criticalmedia.co.uk

Head of Design
Lee Allen

Designers
Sam Ribbits, James Legg, Ty Logan

Photography
Brian O'Halloran

CONTRIBUTORS

Lucy Rogers, Drew Fustini, Jo Hinchliffe, Mayank Sharma, Marc de Vinck, Luke Wren, Rob Miles, Andrew Lewis, Tony Goodhew, Alex Bate

PUBLISHING

Publishing Director
Russell Barnes
russell@raspberrypi.com

Advertising
Charlie Milligan
charlotte.milligan@raspberrypi.com

DISTRIBUTION

Seymour Distribution Ltd
2 East Poultry Ave,
London EC1A 9PT
+44 (0)207 429 4000

SUBSCRIPTIONS

Unit 6, The Enterprise Centre,
Kelvin Lane, Manor Royal,
Crawley, West Sussex, RH10 9PE

To subscribe
01293 312189
hsmag.cc/subscribe

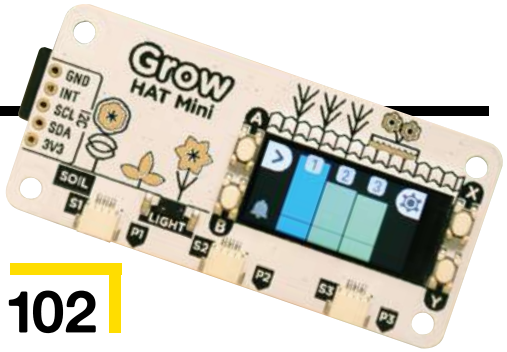
Subscription queries
hackspace@subscriptionhelpline.co.uk

recycle

This magazine is printed on paper sourced from sustainable forests. The printer operates an environmental management system which has been assessed as conforming to ISO 14001.

 ISSN: 2515-5148.

Contents



102

06 SPARK

37 LENS

Cover Feature

38

Tutorial

Laser-cut printing

80

Turn blocks of plywood into your own artisan printing press

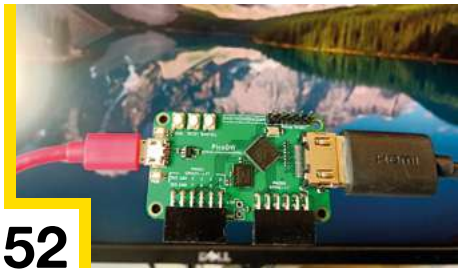
94

Interview

Geeky Faye Art

Direct from Shenzhen

Logic analyser



73 FORGE

101 FIELD TEST

Potion of Healing

By Alex J'rassic

hsmag.cc/Potion

Dungeon Master Alex J'rassic decided to treat her players to this handmade Potion of Healing dice shaker ahead of their new campaign.

For the uninitiated, Dungeons and Dragons is a fantasy tabletop role-playing game, where players work together to complete missions using logic and dice rolls.

When their health is low, usually due to conflict or a poorly cast spell, players need to rest or consume Potions of Healing to regain hit points, and the strength and rarity of the potion dictates the number of four-sided (d4) dice a player rolls.

Part-filled with coloured resin and loaded with dice, the small, glass bottle is complete with a coffee-stained, custom-designed label that contains all the information a player needs to use the potion, and a list of the types of damage their character may receive.

Alex made a video of how she created the Potion of Healing for her YouTube channel (see link above), and you can get your own from her Etsy store (**hsmag.cc/PotionShop**)

Right
Looking at this makes us nostalgic for British childrens' TV programme *Knightmare*


Three Goblins Emporium
Potion of Healing
Greater - 4d4+4
Supreme 10d4+8

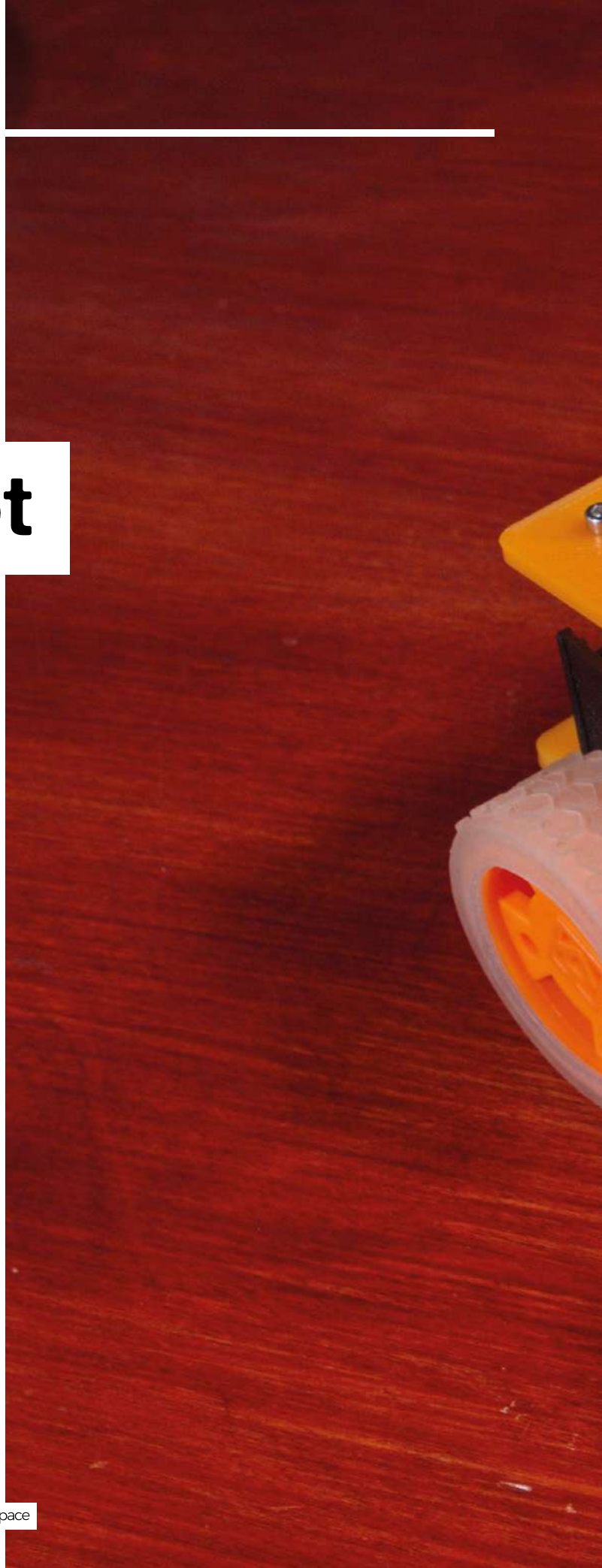
REGULAR

Custom robot

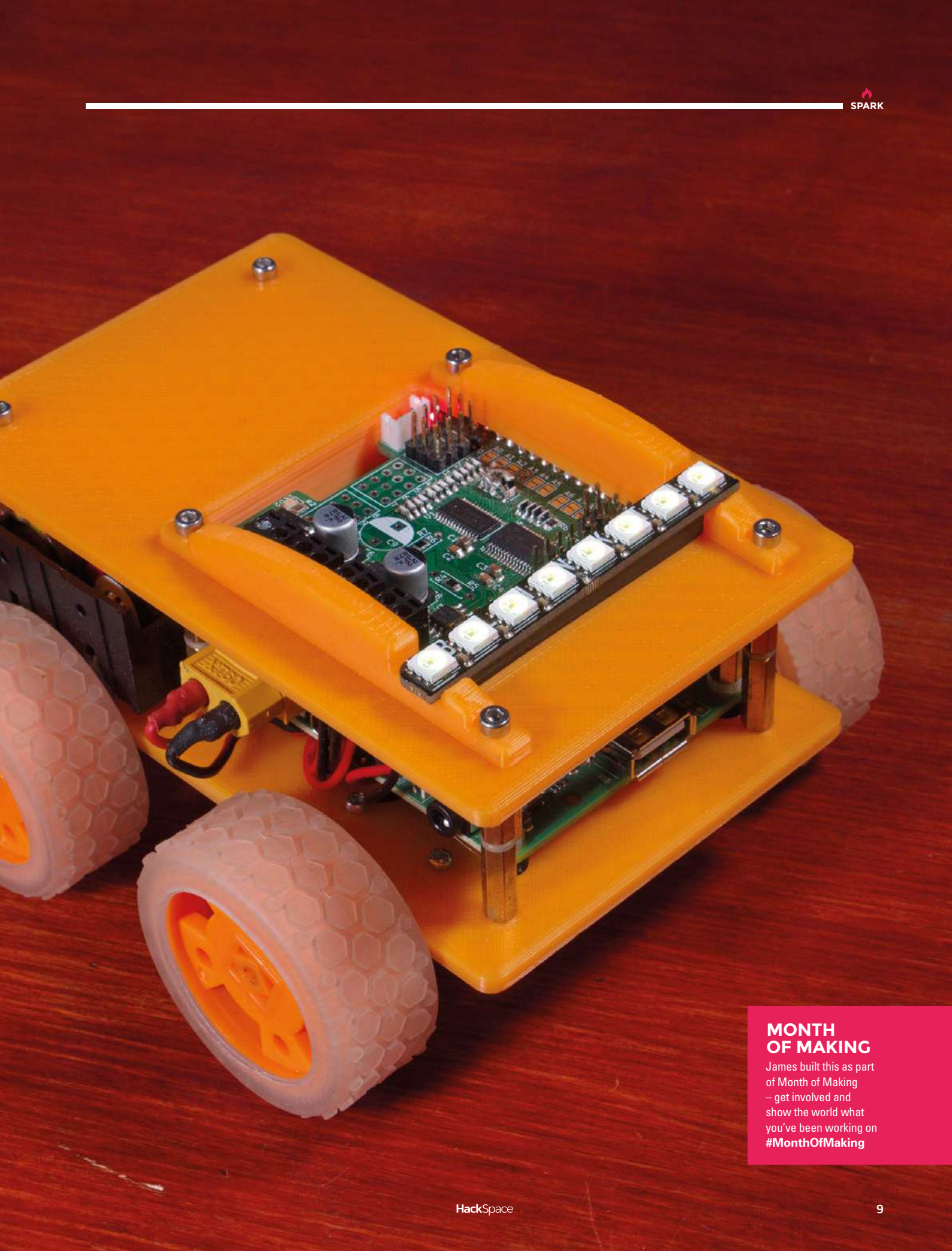
By **Paul Fretwell** hsmag.cc/RobotPiWars

We're all missing face-to-face events (though hopefully not for too much longer). In the meantime, a lot of the things that we would normally look forward to – such as Maker Faires, local hackspace events, and big Hackster conferences – have been forced to go virtual. One of these is Pi Wars.

The event may be virtual, but the robots are still very much real, as typified by this effort from Paul Fretwell. Paul has eschewed the many robot kits on the market in favour of designing his own 3D-printed body including many clever features, and 3D-printed mounting brackets for the motors he's used. It's fast, it's four-wheel drive, and we can't wait to see it in real life.



Right
Keep an eye on hsmag.cc/PiWars2021 for Paul's talk on designing your own robot circuits



MONTH OF MAKING

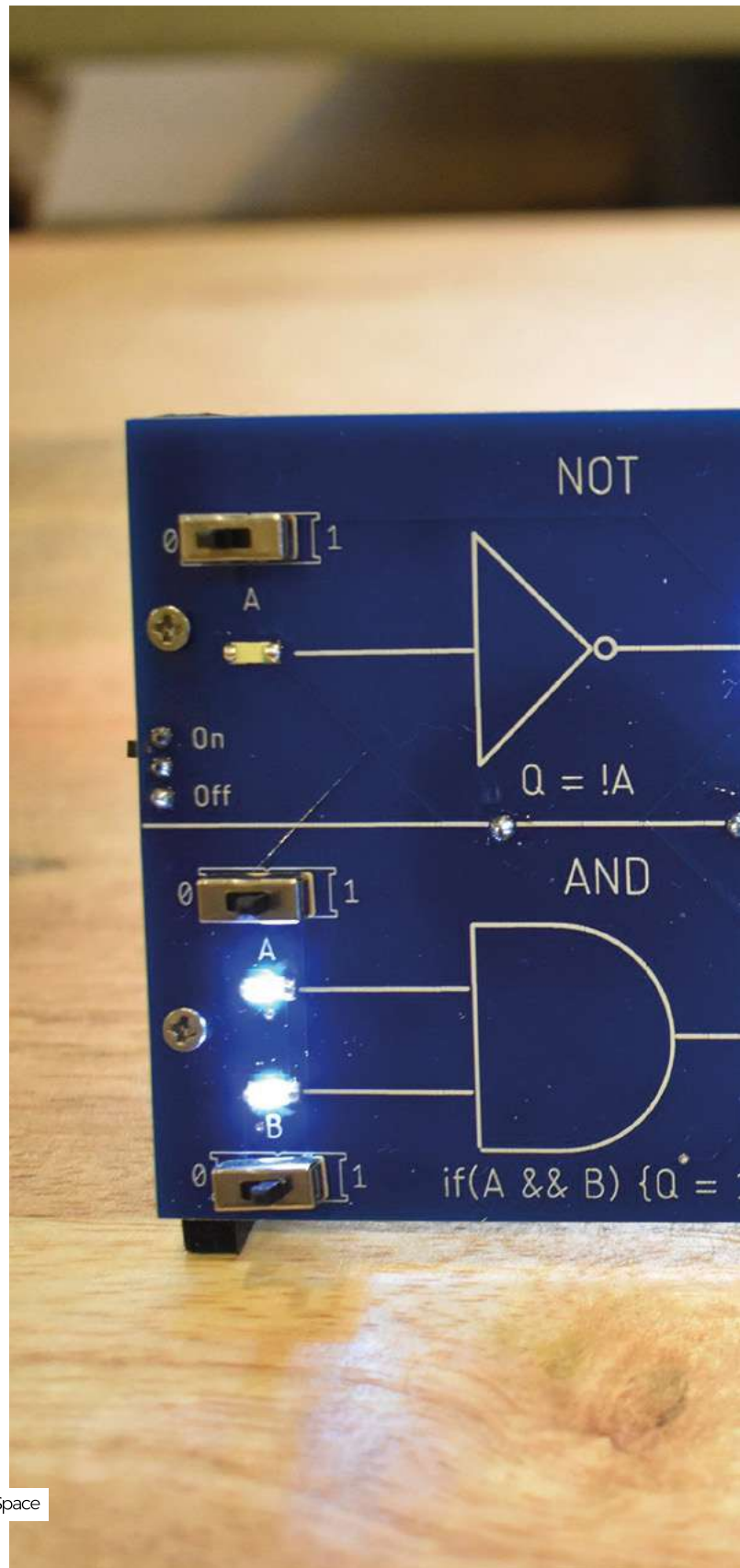
James built this as part of Month of Making – get involved and show the world what you've been working on **#MonthOfMaking**

Logic gates

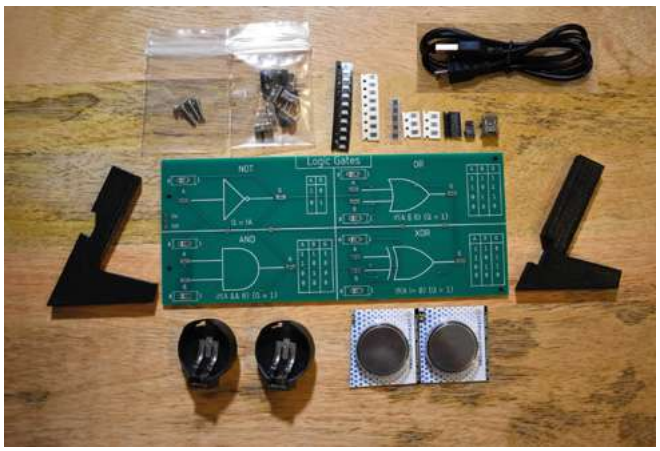
By **TSJ Electronics** hsmag.cc/LogicGate

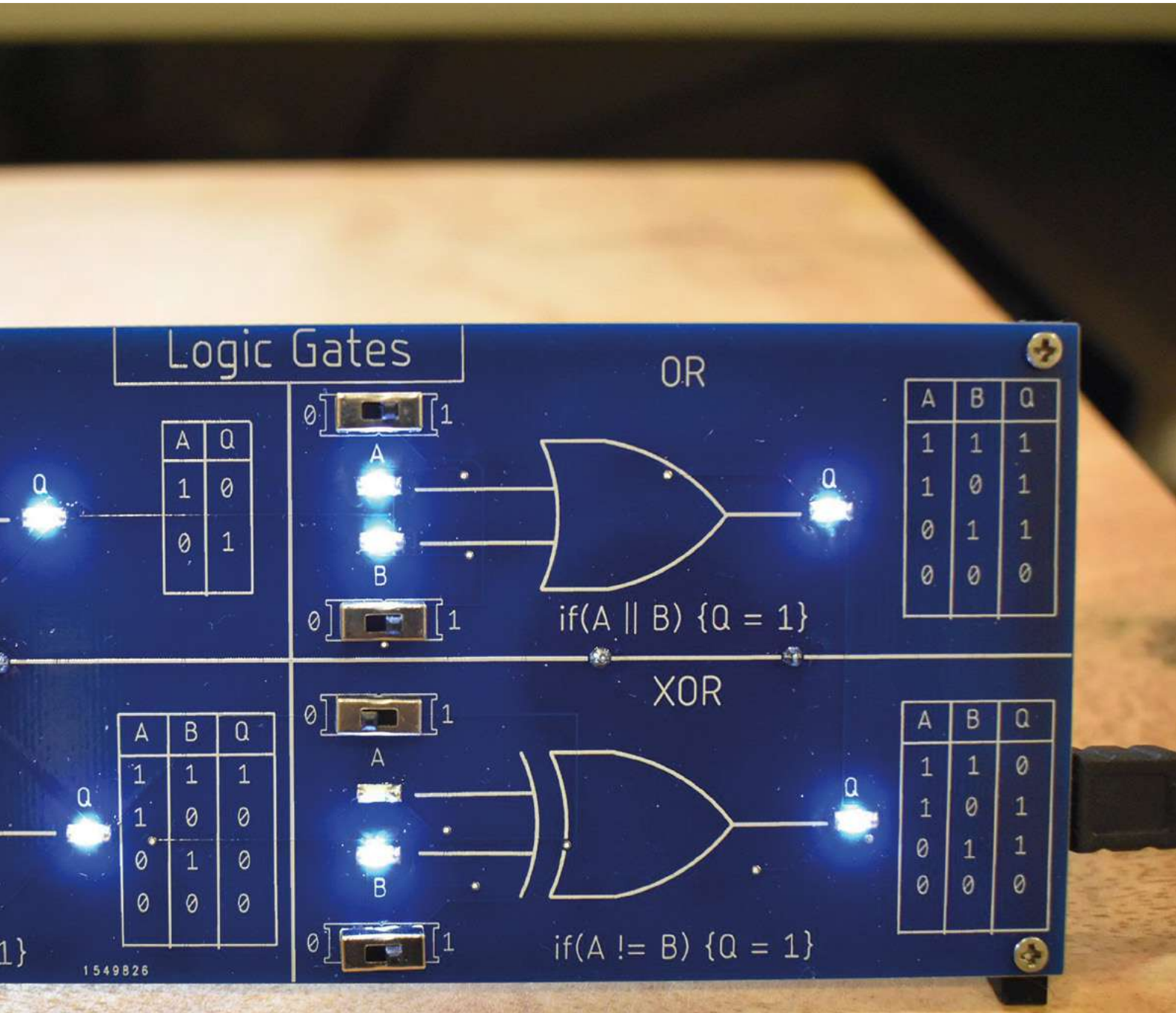
If you came to electronics via any route other than formal education, you might struggle a bit with some of the terms used. AND and OR look like English words, but what on earth is a XOR? And is NAND a real thing, or a typo? None of it makes sense – and as it's electrons, they're too small to see with our puny human eyes. Damn and blast!

But hold on: there's no need to get into a mountain of debt pursuing a degree in electrical engineering. All you need to do is get one of these attractive, interactive logic gates for your desk. It'll teach you the NOT, OR, AND, and XOR logic gates, looks good, and comes with all the parts needed for assembly. Learning by blinkenlights!



Right
Install one of these at home, and learn effortlessly




Logic Gates
OR
A
B
Q
0
1
if(A || B) {Q = 1}
XOR
if(A != B) {Q = 1}
1549826

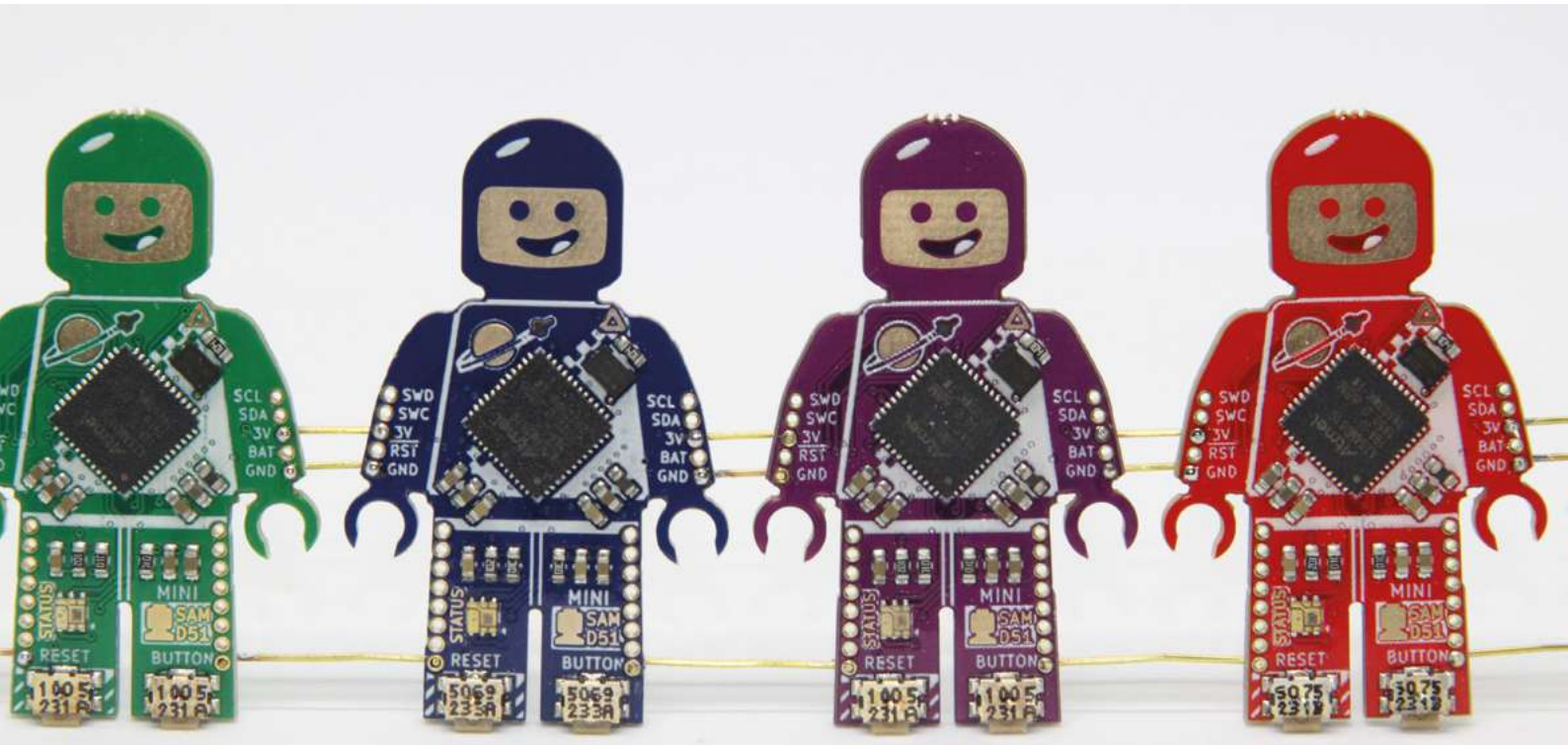
Minifig board

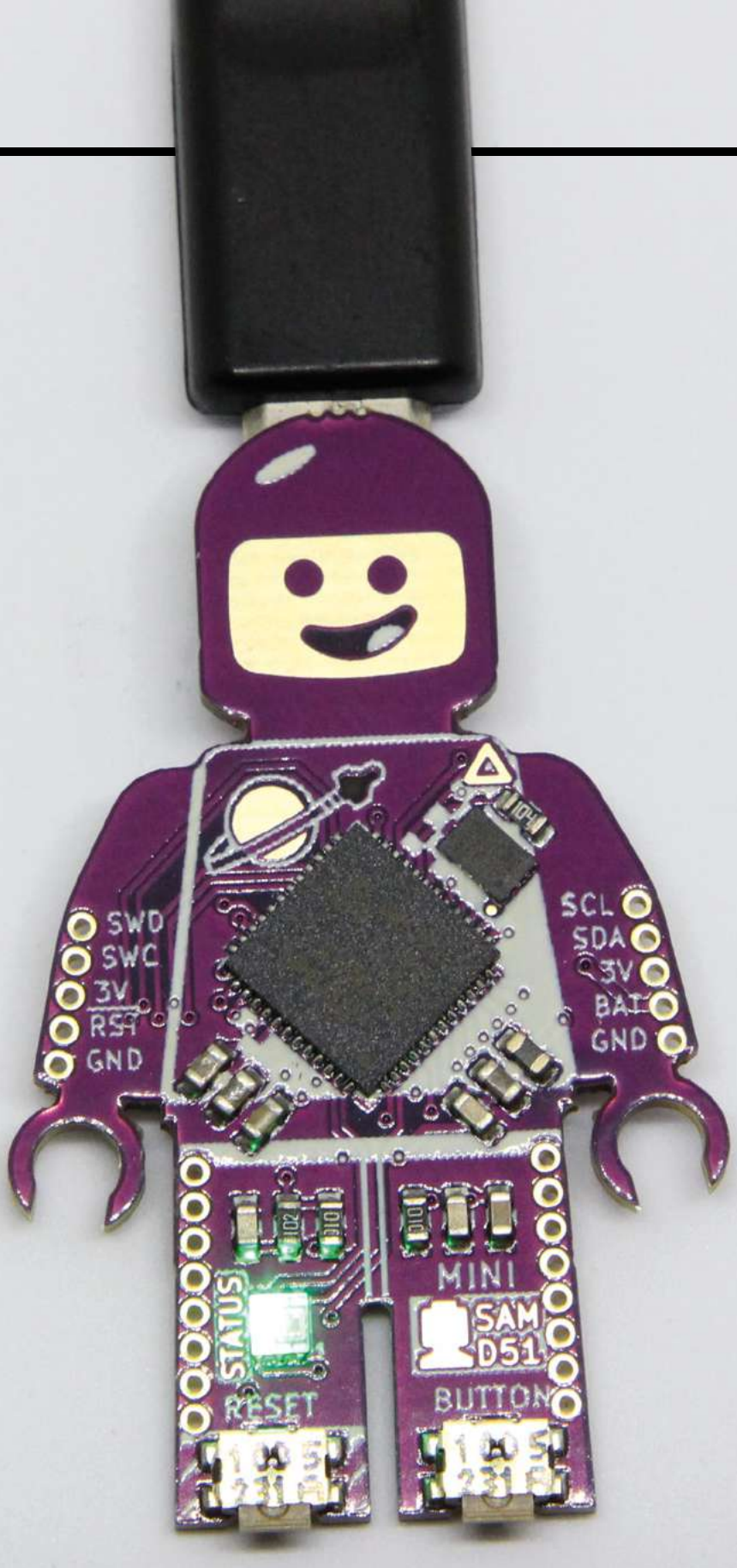
By Benjamin Shockley

minifigboards.com

We've been spending a lot of time looking at Raspberry Pi Pico over the last few weeks, but you don't need to be an established computer manufacturer to take an existing chip and a few components and combine them on a custom PCB… Bingo! You've got a development board. It sounds simple, and in theory, it is, but in practice, it isn't – there's a huge amount about this to get right, which is why we say yay for the likes of SparkFun, Adafruit, et al., who make things for us.

Benjamin Shockley has created his own line of ARM Cortex boards that you can program in either Arduino or CircuitPython. More important than that, they show that when you're not trying to maximise manufacturing efficiencies, you can make boards any shape you want.





Right
So many makers get bitten by the Lego bug – it must have inspired a healthy chunk of global manufacturing

REGULAR

3/4-size arcade cabinet

By James Milroy

hsmag.cc/StarWarsCab

This is James Milroy's ¾-size *Star Wars* upright cabinet, powered by Raspberry Pi and made in March – the Month of Making! A load of techniques have gone into this build, including CAD, CNCing the 18mm MDF for the cabinet, 3D-printing the joystick handles to get that authentic X-wing feel and, of course, connecting all the electronics up.

At the heart of the build there's a Raspberry Pi 3B+ (soon to be upgraded to a Raspberry Pi 4) running MAME, and a Pimoroni Picade X HAT.

Right
The original game was just called Star Wars – not A New Hope or Episode 4 – just Star Wars. And Greedo shot first!

MONTH OF MAKING

James built this as part of Month of Making – get involved and show the world what you've been working on **#MonthOfMaking**

Steampunk keyboard and mouse

By steampunkable

Customretrodesign.com

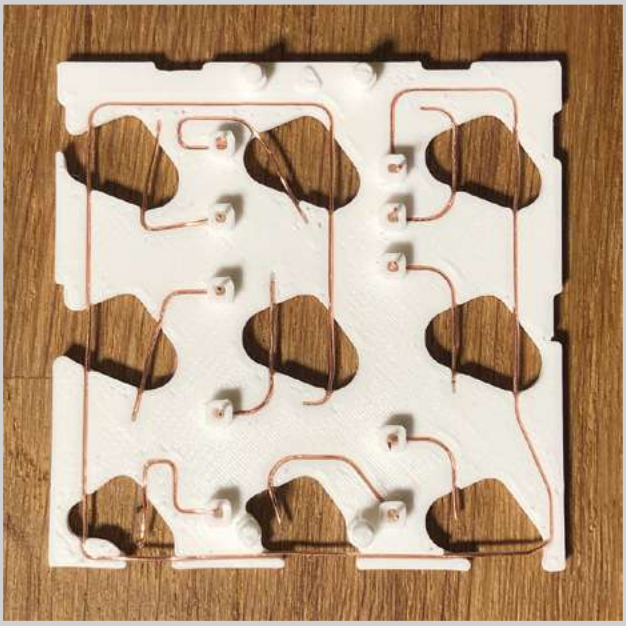
We love the steampunk aesthetic, but it's often not entirely functional. This PC setup, which is not the most ergonomic of things we've seen, is a completely working computer, built for a hotel in Italy.

David, the maker behind steampunkable, put it like this: "Uniquely created builds from keyboards, mice, full setups, towers, as well as high-end hotel reception installations. Not only this, but movie props that have featured in various sci-fi and fantasy productions. Retro tech at its best."

Right
David does custom builds for home, office, TV, and film

Objet 3d'art

3D-printed artwork to bring more beauty into your life



There are many custom keypads out there, but this one is unique: it uses nine mechanical CHERRY MX switches for a chunky, clicky feel, controlled by a Raspberry Pi Pico to interpret these clicks into useful commands.

Instead of a PCB, with its layers of copper, substrate (usually fibreglass), solder mask, and silkscreen, Reddit user duzitbetter 3D-printed a thin layer that holds the Pico in place, as well as the wires that connect to the keys. With just nine keys to wire up, this method makes sense – it's actually a throwback to point-to-point wiring, as seen in high-end audio appliances (if you have an especially high-end guitar amp or an old radio, you most likely own an example of point-to-point wiring).

The keycaps and the enclosure for the device are all 3D-printed, giving this object a lovely lo-fi feel, despite the high-tech nature of what it does. ◻

hsmag.cc/3DPCB

REGULAR

Meet The Maker: Matthew Little – Curious Electric

Electronics to help you monitor and measure your world

In these rather odd times, it's good to have more than one string to your bow. Take Matthew Little, for example: he's a researcher into renewable energy at Loughborough University; he builds pedal-powered electrical installations to show off in schools and festivals; and he's also the proprietor of The Curious Electric Company.

The kits for sale on Curious Electric started life as projects for Nottingham Hackspace (you may remember the Solar 8 Ball Soldering Kit that we featured in our Best of Breed last issue). Rather than pack them away in a drawer and forget about them after the workshop was over, Matthew put them on the internet for the world outside Nottingham to enjoy, and the site has grown from there. We caught up with Matthew to find out what he's up to, what made him start an electrics company, and how he's helping zookeepers keep an ear out for Canadian bats.

> "Lots of my main work has been on renewable energy systems**, and a lot of that work has been on solar power**"

"I don't quite know how I got here, but I've got lots of things that I do – maybe too much! At the moment, you have to be a bit dynamic and change as the situation changes. I've got the electronics kits, which sprang out of helping set up Nottingham Hackspace. They're interesting because I've done them for events and for group learning, and my logic is that I've done them now – why not sell a few if I can?

"I'm also a researcher at Loughborough University, that's my day job, but I only do three or four days a week on that. And then I also do pedal-powered equipment, mainly for events and things, but that's not happening at the moment, so it's great to have the kits as well.

"My main interest is in electrical systems and renewable electricity. That's my main focus at Loughborough. A lot of my work to get here has been on monitoring renewable energy systems, installing renewable energy systems in the UK and in other countries. I also teach a course on off-grid power supplies.

"It all stems back to being given a solar panel while I was working on an undergraduate project. I was always interested in electronics, since I was young. I was given a solar panel for a project and I was utterly amazed by it. I'd seen them before, but having this quite big solar panel at the time – it must have been the mid-nineties – I was fascinated by how it was silent and just did its thing. That made me go down a certain route to look at renewable energy. Lots of my main work has been on renewable energy systems, and a lot of that work has been on solar power.

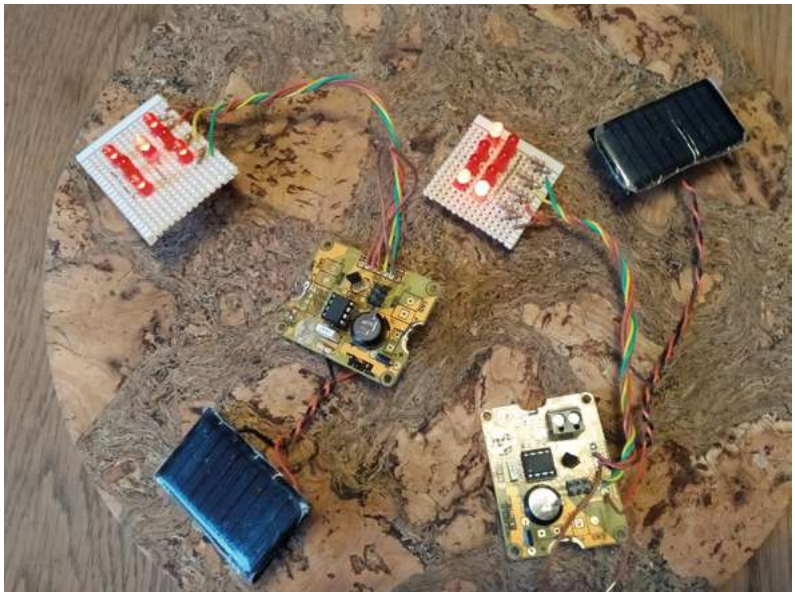
"And then I've always been interested in making electronics circuits. I got involved with Nottingham Hackspace, which has led to a lot of the kits. The pedal power equipment all comes from… I was →

Above
The Curious Electric Company's headquarters – much experimentation takes place here

Right
Building the original version of the Bat Detector at Cambridge Makespace

Below
As solar power gets cheaper, renewable projects become more and more accessible

building these renewable systems, but people didn't really understand about power and energy. The amount of power you use is quite abstract to a lot of people – you just plug in and switch on your device. If you've got a pedal generator, you can actually feel how much you're generating. You can show straight away with different kinds of light bulbs; you can feel there's a difference.



"The renewable energy stuff led me to do demonstrations of pedal power, which has now turned into quite a few projects on that. Electronics is the thread that's run through my career, which has turned into renewable energy.

"We used to do an event where we'd go to schools and we'd get the biggest kid and try to get them to power a little incandescent light, and it would just about be flickering. Then we'd get the smallest kid and get them to light up an LED. It's a nice way of showing things.

"I like positive messages around what we can do relating to the environment. I think pedal power is quite a nice one. Certainly with sound systems, it's amazing how much of an experience you get. A 30–40 watt sound system is incredibly loud, and you can easily pedal that. I like those positive experiences where people have a go and think, 'Wow, this is amazing!'.

"I'm not very good at knowing the target audience for my kits. I'm kind of making them for what I would have liked when I was younger.

"I want them to be accessible to (probably not really young) kids, but I'm trying to inspire people to try soldering.

"A lot of this is on the beginners' side of things. But I think they work as well for people who want to learn how to solder, or just get a bit of experience with electronics. There are loads of people making

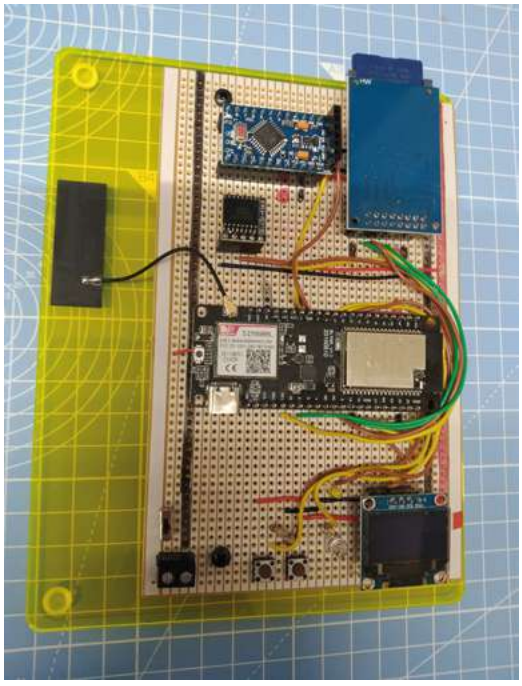
kits, and there are loads of really great ones. I wanted to do something very slightly different and make things as accessible as I could – something you could do at a Maker Faire or at a hackspace as a group project. Nearly all of them have come from that – typically, I've done a prototype and then we've run a course at Nottingham Hackspace.

"I was actually working on a project to do energy monitoring and I heard about London Hackspace. This was probably about 2010. I thought, 'Amazing, this is great! I'll find the one in Nottingham'.

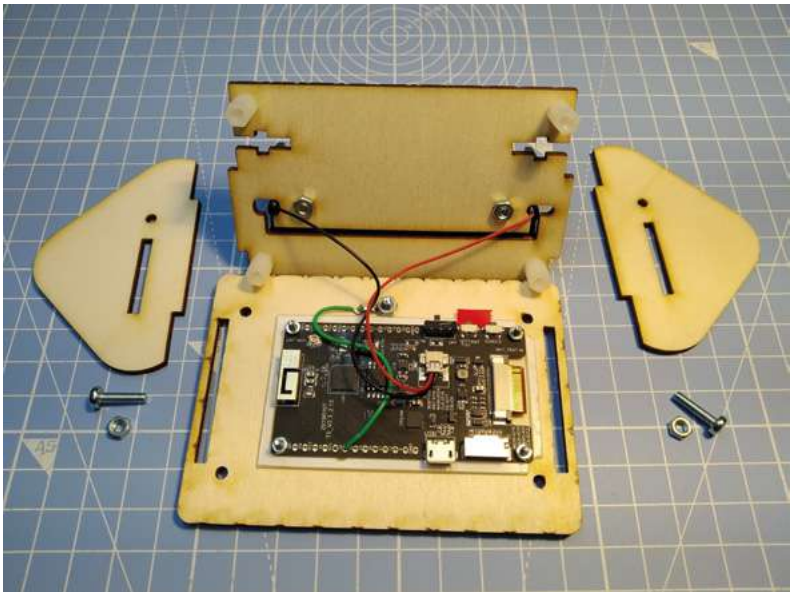
> **We started to get more members, and** then we decided to expand a bit more

"So I did a search and found one in Nottingham – amazing! Clicked on the link to say 'Hi' and it was just another person who two weeks before had posted saying 'Hi, I'm starting a hackspace.' That was Dominic Morrow, **@ChickenGrylls** on Twitter. From that to meeting him – by the time we met up in the real world, there were about five of us to start with. We met up in a pub. I've always had a little workshop, usually in an arts space, because it's been cheap. I was in a relatively cheap building that had been done as arts units, so it was quite DIY. I found out about a unit coming up, and we got that for the Hackspace. We started to get more members, and then we decided to expand a bit more, and we did a crazy thing and got a huge place that was right at the top of our budget.

"My main interest in the Hackspace has always been doing workshops, sharing skills. I've also learned so much from the people I've met. Things like laser cutters I'd never used; I started to use the one at the Hackspace, started to use the CNC machine, and I've been taught to weld at Nottingham Hackspace. →



REGULAR



Right
Solar drag racers are an immediate way of translating light into motion

Below
Many of Matt's projects involve monitoring the world around us

"I've used it a lot for those things that I'm not going to fully invest in. But, when I need them, they're really useful to have. I have a decent workshop. I actually have a laser cutter now. But for ten years I didn't and it's been absolutely amazing, both for access to tools and for the community and some of the people I've met who have been instrumental in developing my work, and also realising that there are other people who are interested in the same things that I'm interested in. That's been the biggest aspect.

"I started selling kits probably about six years ago. The first kit I sold was a bat detector, which takes the ultrasonic frequencies formats and drops them down to human frequencies. I built one for my dad, and thought I'd do a few more and run it as a workshop at the Hackspace.

"I did that at Derby Makers, and at FizzPOP Makerspace in Birmingham as well. I thought I'd try packing 20 of them up and put a little PayPal link on my website, and see what people think.

"It was probably about three years ago that I set up The Curious Electric Company. I decided to do quite a lot of kits for different reasons, and I wanted to have a brand. And also, I wanted all my kits to have a focus – which is looking at the environment, at the world around you, or renewable energy. Nearly all of the kits have something to do with looking around you, measuring or monitoring something. That's the niche that I'm interested in. That, and trying to live as lightly as possible.

"With the kits, I want people to have an experience building it, but I think once it's built, I also want it to be an experience when you use it. I work with microcontrollers. I build and develop on those, but with my kits, I want you to have something that works; something that does a function. I know you can start to program things and do lots more, but I wanted something that you maybe don't need to program; something that you can use as-is in the real world. That's different to quite a lot of other people supplying electronics parts and components.

GREEN SOLDERING IRONS

"I did a monitoring system for air quality monitoring based on the Luftdaten set of air sensors (I think it's called Sensor.Community now). It's based on the ESP8266 with the little air quality particulate matter sensor. I thought it was brilliant. I made a little box, made it waterproof, and I'm hoping to supply a soldered kit to the Nottingham Air Quality group, and help out with getting it set up.

"The other thing I've been involved with is Nottingham Fixers. About six or seven years ago, we set up Nottingham Fixers from the Hackspace, running a little market stall where we could help

people out with their broken electronics. It was difficult; it's quite intense because you're working on people's equipment. You're not a repair shop; you're trying to get them to be part of it, and some people really don't want to fix their own things, or maybe they don't have the confidence to try it.

"We were still figuring out how to do things. That's one thing. Where do we get parts from to replace parts? Who pays for that? The other thing is that when you repair something, you take on responsibility for it being safe. On the very first one, we had a guy who'd heard about it on the radio. He was 70-odd years old. He'd come on two buses, and brought an iron that wasn't working. He'd lived in the area for so long, he was telling us stories about the market area from years ago, and he was so pleased to get his iron fixed. I had to go to Maplins to get some parts, and it took about two hours to fix this thing, but it was great to see someone like that who was so happy to be part of it.

"Everything that I produce for Curious Electric is open-source hardware. Nearly everything in the shop is something that I've designed and made. There are a few things that I buy to resell because they're interesting, and I was trying to have a better range of things. There's an oscilloscope kit that's not open-source. But all of my designs, I aim to be open-source and have the design files out there, usually in a GitHub repository. I use KiCad for all the circuit design. I use Inkscape. Nearly all of the code is done on the Arduino IDE for upload, even if it's not for Arduino. I really want everything to be open. I want people to take it and put a twist on it and do a new thing – it's really exciting when you see people adding bits to your equipment. It's a win-win. I've learned so much from other people's projects – why wouldn't I make my own open-source?"

Left
The whole point of Matt's work is to show people the possibilities of solar and renewable energy

Below
Like an Irish theme pub, Matt's workshop has a bike hanging from the wall

Outside of my comfort zone

Farewell, dear reader

Lucy Rogers

@DrLucyRogers

Lucy is a maker, an engineer, and a problem-solver. She is adept at bringing ideas to life. She is one of the cheerleaders for the maker industry, and is Maker-in-Chief for the Guild of Makers: **guildofmakers.org**

I am a firm believer that the magic happens outside of my comfort zone. When HackSpace magazine (thanks, Ben Everard!) asked me over two years ago to write a column for the then-new magazine, I was delighted. And nervous. Most of my writing until then had been translating science into plain English. It had to be right. But a regular column is (well, this one has been) more of a random thought, feeling, or even, shock horror, my opinion.

At school, I veered away from subjects that required opinions and personal interpretations – I never could work out how they were marked – surely my interpretation was correct – even if it wasn't the same one the teacher had? See also, art appreciation.

So writing about my thoughts in public, even if it was going to be read by the friendly maker community, was daunting.

Who would want to read my ponderings about making? Why would that be of interest? Would it be a page people would skip, before they got to the 'How to' pages, or the interviews?

These questions I kept in mind when writing these columns. For my own maker projects, I owe much to those around me who have passed on their knowledge and experience.

Therefore, in writing, I often tried to pass on something I have learnt that may be of use to someone else. I have been using this column, in a little way, to pay forward some of the help I have received.

I am sure we have all been in the situation where the storyteller starts to repeat themselves. I remember, back in the days of being able to go to the local pub, this would often happen at around half an hour before closing.

> **I have been using this column, in a little way, to pay forward some of the help I have received**

New experiences take longer to acquire than writing about them. I feel I have already shared my best stories! There also comes a time when my comfort zone expands, and what was new, scary, and uncomfortable becomes routine. And, for me, when that happens, I know I do not put the effort, care, or commitment in that I used to. Hopefully, the occasional piece of magic happened for you in the process.

But now it is time for me to pass this column on to a fresh new voice. One that can entertain and interest you and maybe help you think in a slightly different way.

So thank you, dear reader, for coming along on my journey – and making it magical.

Happy making!

Car hacking

Driving safely

Drew Fustini

@pdp7

Drew Fustini is a hardware designer and embedded Linux developer. He is the Vice President of the Open Source Hardware Association, and a board member of the BeagleBoard.org Foundation. Drew designs circuit boards for OSH Park, a PCB manufacturing service, and maintains the Adafruit BeagleBone Python library.

Hacking a car in the movies looks easy: you poke around a bit with a screwdriver, you twist together some bare wires, and you're good to go. Happily, the security systems on modern cars are a bit more sophisticated than that, but with the right skills and tools, hackers can exploit all sorts of weaknesses to gain unauthorised access to a car: on-board computers, key fobs, Bluetooth connections, and even the pressure sensors on your tyres.

These weaknesses can be extremely dangerous, and car manufacturers can get very sloppy with their security. In just one example, a car hacker known as L&M realised that two GPS car tracking apps had given all customers the same default password (123456) on sign up, allowing anyone to gain access to thousands of accounts. As well as the flaws providing access to personal and financial details, L&M exposed an incredibly dangerous vulnerability: the ability to remotely stop the engine of some of the vehicles using these apps.

> **Hackers can exploit all sorts of weaknesses to gain unauthorised access to a car**

Finding and reporting vulnerabilities like this is central to the car hacking community. There has been a lively, Car Hacking Village at DEF CON since 2015, where car hackers educate security researchers about modern-day vehicle systems, experiment with technology, and play with all sorts of motorised vehicles, from upgrading mobility scooters to making a car escape room where you have to hack your way out of a locked SUV.

The best way to get started with car hacking is to get yourself a copy of *The Car Hacker's Handbook* by Craig Smith of **@OpenGarages** and a car hacking board such as the CANtact, the M2 by Macchina, or the Carloop. These devices plug into the OBD-II diagnostics port, standard in all vehicles made in the last 25 years, and communicate over the CAN bus with the ECU (Engine Control Unit) and other sensors and actuators throughout the vehicle. Open-source programs exist to both interpret the messages on the CAN bus, like tachometer data, and send messages to control dashboard readouts and much more.

There are also lots of car hackers on Twitter that I follow to keep up with the latest news, including Robert Leale (**@carfucar**) and Kirsten Sireci Renner (**@Krenner**), who co-founded the DEF CON Car Hacking Village, and Ian Tabor (**@mintynet**), who runs the UK Car Hacking Village.

Letters

ATTENTION ALL MAKERS!

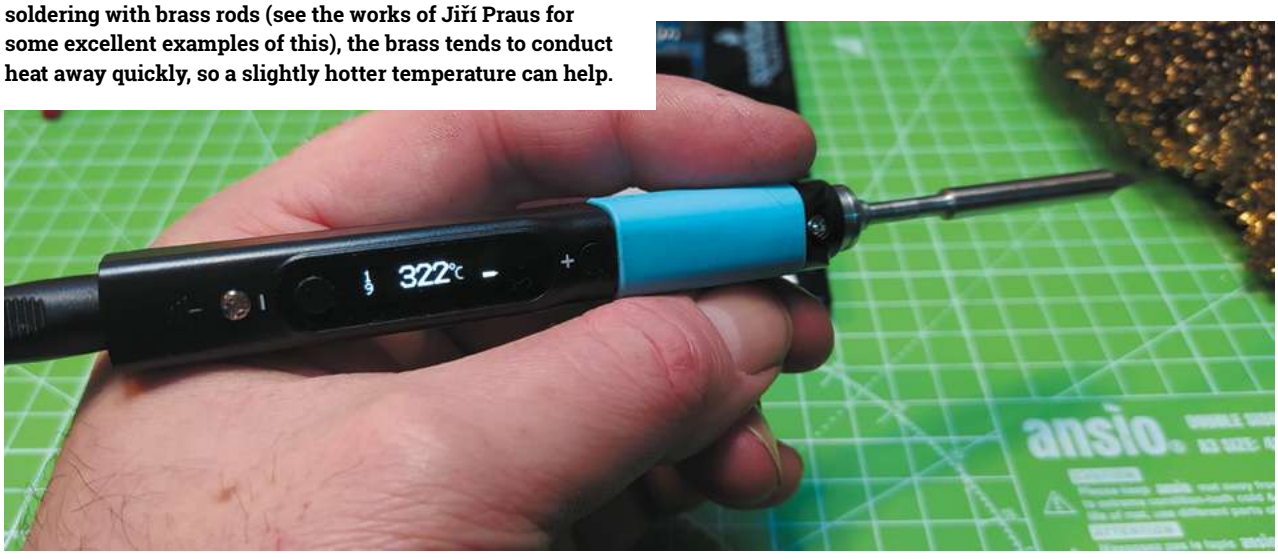
If you have something you'd like to get off your chest (or even throw a word of praise in our direction) let us know at **hsmag.cc/hello**

SMART SOLDERING IRON

I may be getting a bit slow in my old age, but what on earth is the point of a smart soldering iron? There's a cool end, which you hold, and a hot end, which melts things. Surely there can't be much more to it than that?

Adam
Dublin (Ohio)

Ben says: In essence, there isn't much more to it than that. If you're happy with what you're using right now, stick with it, and good on you. But a smart soldering iron (or a soldering station, if you have space) gives you more control over the temperature your iron reaches. So, if you find yourself damaging PCBs or components, you might benefit from using a lower temperature. Likewise, when you're free-form soldering with brass rods (see the works of Jiří Praus for some excellent examples of this), the brass tends to conduct heat away quickly, so a slightly hotter temperature can help.



WINTERBLOOM

After reading Meet the Maker with Thea Flowers last issue, I've gone down the rabbit hole of the Roland JUNO synth. Annie Lennox – what a voice! And the synths – incredible! I want one. Not the 1980s version – the modern, improved Winterbloom version. TAKE MY MONEY!!

Kelvin
London

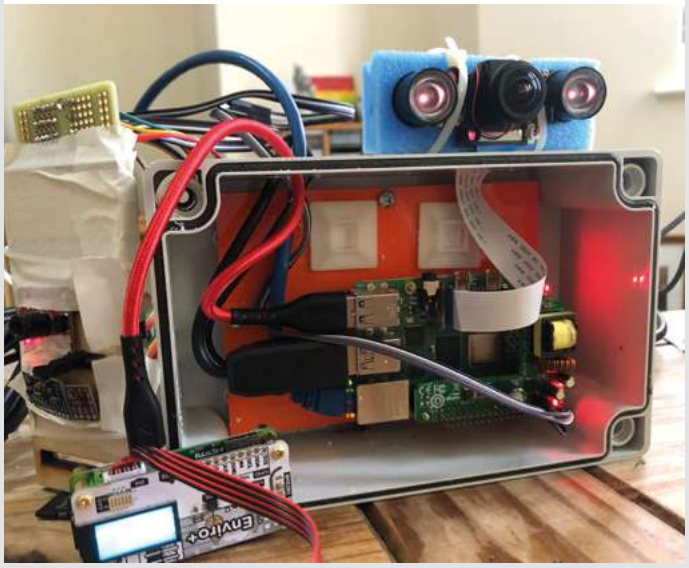
Ben says: We've touched on the world of modular synths in HackSpace magazine before in issue 14, where we walked through building a DIY synth. That was pretty involved, and even then it produced something incredibly basic. So for Thea to take her work and make all of it open-source, for anyone to learn from, is incredibly public-spirited.

BEES

It's always chafed on me a bit that once you leave school, unless you're a scientist, you don't get to do science any more. Any amateur can write a song or paint a picture, and if it's rubbish, it doesn't matter, as long as you're having fun. But science always seemed like something I've cut myself off from after not choosing a science degree. I read your interview with the bee-keepers, and how they choose the kit that they use – it's all affordable stuff that I can buy off the shelf and that I can understand! (Maybe not understand right away, but I can teach myself). Now, bring on the spring and let's monitor the bees!

David
St Albans

Ben says: There are many things wrong with the world and his dog having access to the internet, but one of the good things about it is that scientists can work together on large shared datasets. The barrier to entry has never been lower if you want to get stuck in and share with research, as projects like Folding@home and Sensor.Community (formerly known as Luftdaten) show.



CORRECTION

In issue 40, we published a tutorial about crocheting a resistor-holder written by Anuradha Reddy. We didn't include a picture of Anuradha with her creation. Sorry about that Anuradha.

CROWDFUNDING NOW

Plybot

The 3D printer with arms

From $299 | **kickstarter.com** | **Delivery:** from July 2021

Plybot – so named because it was originally designed to be flat-packed and made of plywood – is a 3D printer with quite an unusual design. While the Z-axis is similar to most 3D printers, the X and Y are controlled by arms in a way that's reminiscent of (but not the same as) delta printers.

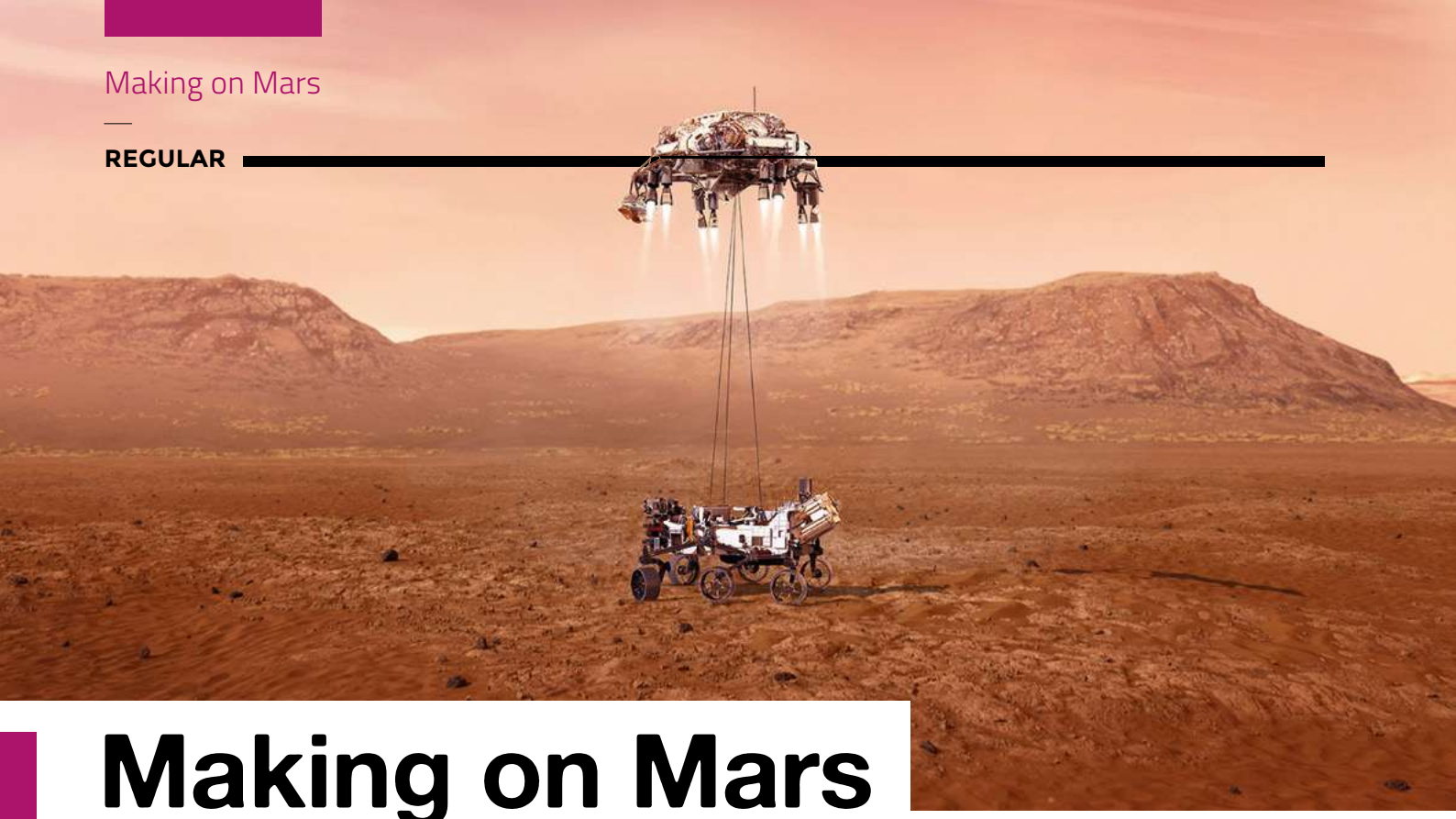
Honestly, we don't have any idea how this will perform. It's unlike anything we've used and we're excited to find out if this setup will prove to be good or not. The makers claim that the arms arrangement lets Plybot print bigger, faster, and more reliably. The first point looks valid – at least from the images we've seen, the ratio of print bed size to printer desk footprint looks better than most alternatives. Because the arms flex in the middle, they don't need long run-outs.

The joints in the arms look like weak points. However, with two arms attached to the print head, that may mitigate this. And even if they are weak points, it doesn't mean that they can't be made strong enough by some thoughtful engineering choices.

We'd love to give some thoughtful analysis on how well we think this will work, but the truth is that we're waiting to find out. It's exciting to see people bringing new 3D printer concepts to market.

BUYER BEWARE

When backing a crowdfunding campaign, you are not purchasing a finished product, but supporting a project working on something new. There is a very real chance that the product will never ship and you'll lose your money. It's a great way to support projects you like and get some cheap hardware in the process, but if you use it purely as a chance to snag cheap stuff, you may find that you get burned.



Making on Mars

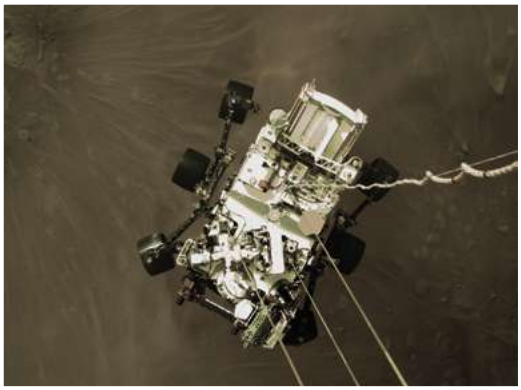
How makers are helping in the search for life on Mars

It's been a good few weeks for space. We've had the astonishing footage of SpaceX's latest vertical take off and landing attempt. Less flashy, but far more exciting for David Bowie fans, was the successful landing and first images beamed back from the Perseverance rover on Mars.

We've sent wheeled and tracked rovers to the moon and to Mars before, and as technology gets better, the images we get beamed back to us on Earth get correspondingly better. This time though, NASA is aiming for a true tech first: the first flight on another planet.

Attached to the under-belly of the Perseverance rover is the Ingenuity Helicopter, the tiny (it only weighs 1.8 kg) flying machine that's going to answer the question of whether it's possible to fly in the extremely thin atmosphere of Mars. Ingenuity has two 1.2 m diameter coaxial rotors, spinning in opposite directions on the same plane. It will have a maximum speed of 10 m/s, and will fly at between three and five metres above the surface of Mars.

Payload is a camera, facing down toward the red planet's surface. All being well, the images that the helicopter sends back will reveal more detail on Mars' terrain, including landing sites, and perhaps even little green men/microbes. Space being space, and engineering being engineering, NASA isn't expecting great amounts of data from the camera; instead, they're more interested in learning what



Left
According to SparkFun, the LIDAR-Lite v3 is the ideal solution for drone, robot, or unmanned vehicle applications

they can about flight – if it's even possible, for one; whether a coaxial helicopter is the best approach, and whether they've made the right assumptions about the effects of Martian gravity and atmosphere on flight characteristics. Although the gravity on Mars is only about one-third the strength that it is on Earth, the atmosphere is a lot thinner – the surface

> **The images that the helicopter sends back will** reveal more detail on Mars' terrain

density of Mars is only 1% that of earth, meaning that it's equivalent to flying 30,000 feet above the surface of Earth.

THE MAKER CONNECTION

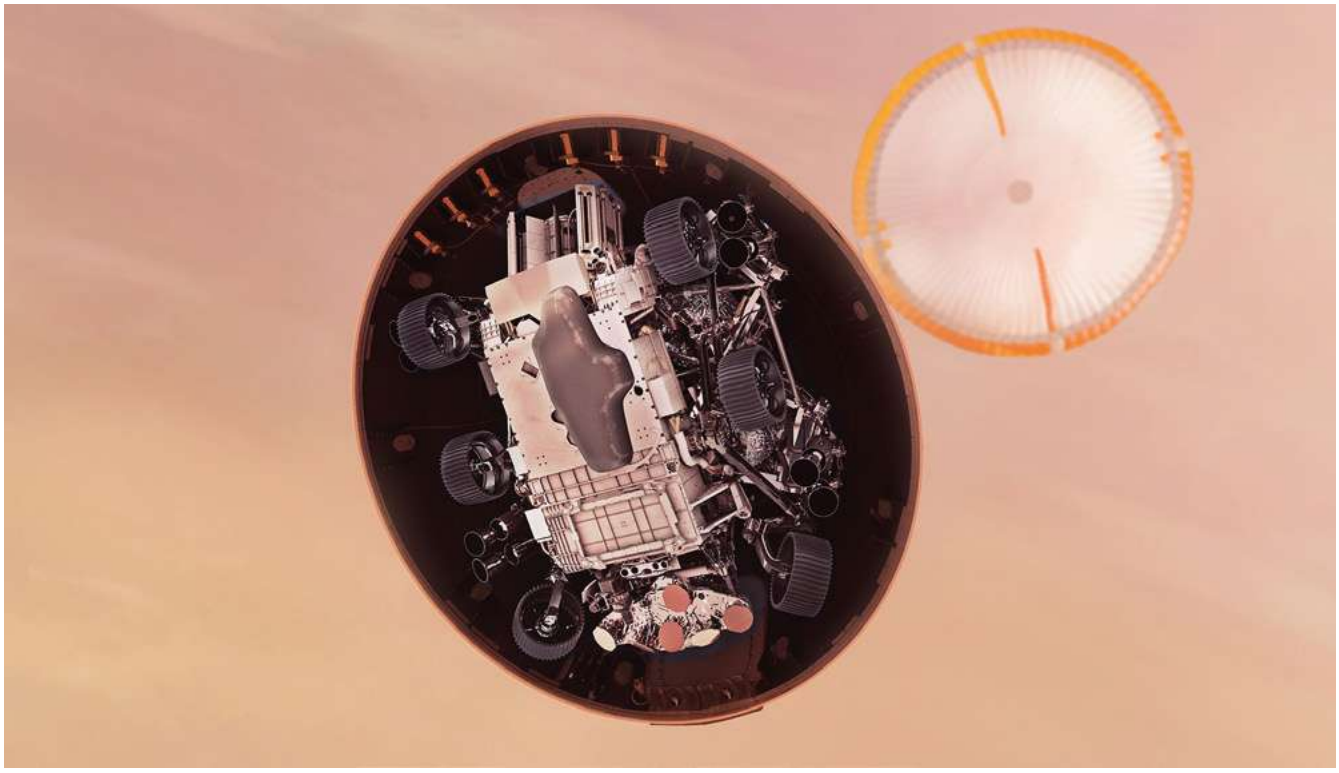
This is all great stuff, but what does it have to do with making? Well, the helicopter is a proof of concept, so NASA didn't have tons of money to throw at it. Instead, they sourced as much as possible off the shelf – including the LIDAR-Lite v3 laser altimeter bought from SparkFun, for the princely sum of $130. For context, that's 0.00016% of Ingenuity's $80 m development budget. →

REGULAR

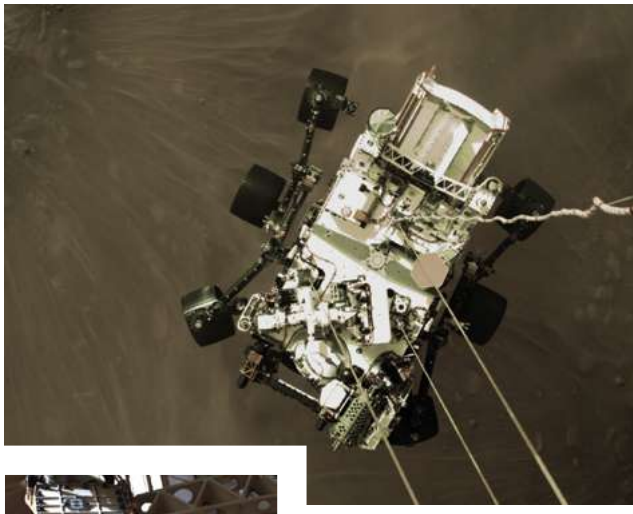
Right
Mars Yard is in Southern California – generally an easier place to work than the surface of Mars

Off-the-shelf components have two advantages over custom-made ones: the first, quite obviously, is the price. The second is less apparent, but it's that they can be far more powerful than the equivalent custom piece. Big budgets take a long time to approve before any work gets done, so in the time it takes a branch of the US federal government to fund, commission, design, and build a given piece of hardware, the limits of what's possible will inevitably have moved on.

Space age technology is all around us – Ingenuity uses a Qualcomm Snapdragon 801 processor – the same chip that's used in smartphones, including the Google Pixel 4a. And it runs Linux, which is free to install whatever your budget. This is a truly fantastic time to be a maker – the toy box available to us has never been better. NASA certainly think so, and who are we to argue with that? ◻



> This is a truly fantastic time to be a maker – **the toy box available to us has never been better**



Build a Makerspace for Young People

Join our **free online training course on makerspace design** to get expert advice for setting up a makerspace in your school or community.

Sign up today: rpf.io/makerspace

HackSpace
TECHNOLOGY IN YOUR HANDS

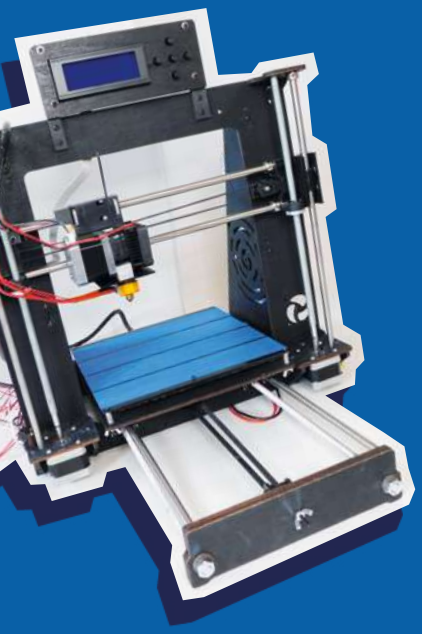
LENS

HACK | MAKE | BUILD | CREATE

Uncover the technology that's powering the future

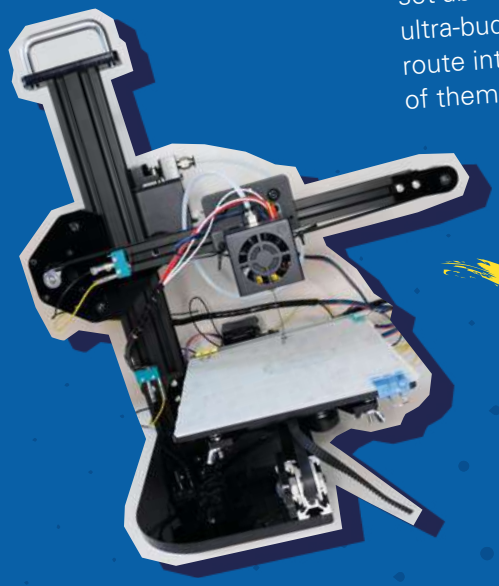
ULTRA BUDGET 3D PRINTING

CAN YOU GET A WORKING 3D PRINTER FOR UNDER £100?



How much does a 3D printer cost? There's no easy answer to that question. They can cost thousands of pounds. They can cost hundreds of pounds, but what about less than that? We took to the internet to find the best printers we could get for under £100 and put them to the test. Most of them cost around £70–£80 (that's around $100 for our readers across the pond). For that money, we wanted everything we needed to get started – a complete printer.

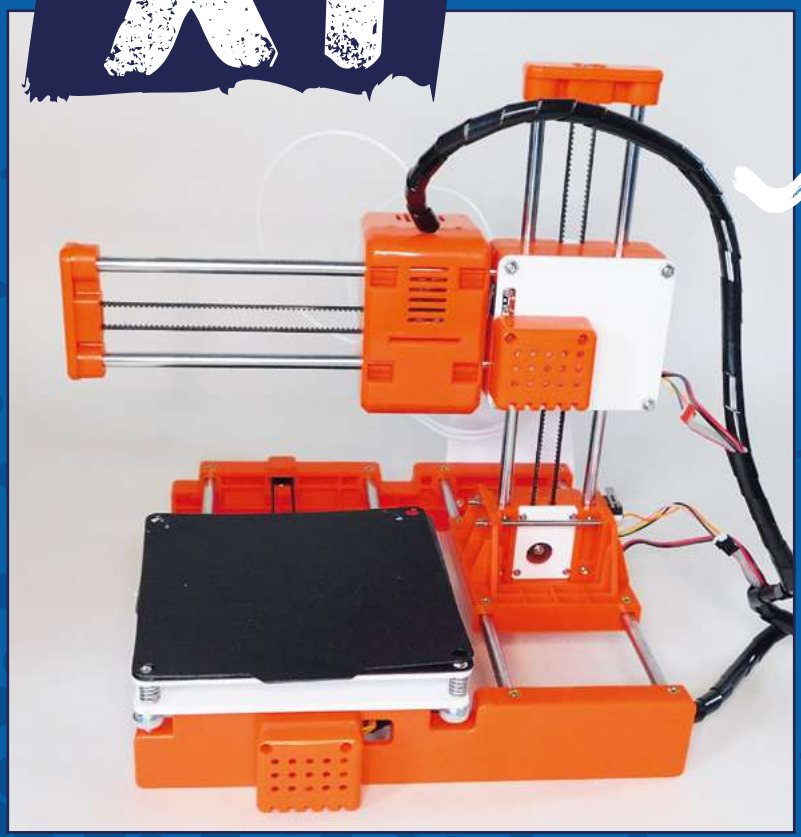
Just a few years ago, this would have been an impossible task unless you wanted to repurpose some bits scavenged from broken equipment and build yourself a fabricator. These days there are multiple companies that claim to provide everything you need for this cost, but do these ultra-budget printers really work? We bought five printers and set about finding out what you can expect from an ultra-budget 3D printer and if there is a possible route into 3D printing in this price range. Will any of them work? Let's find out. →



EASY THREED X1

CLICK, CONNECT, AND PRINT

We paid £69.13



This printer comes in two parts. Slot the top onto the bottom, screw in two screws, attach one wire, and you're ready to print. It really is that easy. The only problem we had came when we tried to load the filament. And try as we might, we couldn't get it to load. Looking at the internet, this isn't an uncommon problem. The solution to it, however, is very straightforward. The sample PLA comes tightly wound, and the print head doesn't like filament with much of a curve in it. Straighten it out and it slots in.

There's not much to the printer controls on this. A button to home the printer, two buttons to feed or retract the filament, and one to print. That's it, but it does give you just enough control to do what you need to do – load and unload filament, level the bed, and print. The slightly upgraded X2 model includes an LCD and more of a user interface.

Once we'd managed to load the filament, everything else was straightforward. The standard four screws levelled the bed (removable and magnetic), and it was ready to go. We were pleasantly surprised by the quality of the prints. The test prints that came with the printer turned out well. We had no issues with printing – just press the button and prints come out.

Right
The separate control box is spartan, but has the essentials

SAFETY

3D printers have to heat plastic hot enough to melt it, and this may not be that much below the point at which the filament catches fire. It's also shuffling reasonably high currents around to do this. There's always a risk of fire with 3D printers, and ultra-budget machines may not have been made with the care of some of their more upmarket cousins. We strongly recommend that you don't leave these printers running unattended.

FILAMENT QUALITY

Most 3D printers come with a small amount of filament with which to test them out. This filament is usually of awful quality. 3D printer filament absorbs moisture over time and this moisture turns to steam when it's heated up. You may hear little pops coming from the extruder as you print – this is caused by bubbles of steam popping. It's probably no surprise that it's hard for your 3D printer to print well when bubbles are popping in the filament. If you have trouble with stringing, or it looks like there are bits of filament missing in your print, the problem likely is with the filament rather than the printer.

Above
These test prints show that the X1 can do reasonable quality prints in its small print volume

One of the problems with building a 3D printer is making the frame stiff enough. Any slight wobbles show up as imperfections on the print. This frame, as you can probably see, isn't the most robust. To a certain extent, the small print volume mitigates this – shorter axes mean that there's less leverage to each wobble – but there's still noticeable ghosting on sharp corners. This isn't a particular problem as it's mostly aesthetic, but it is something to be aware of.

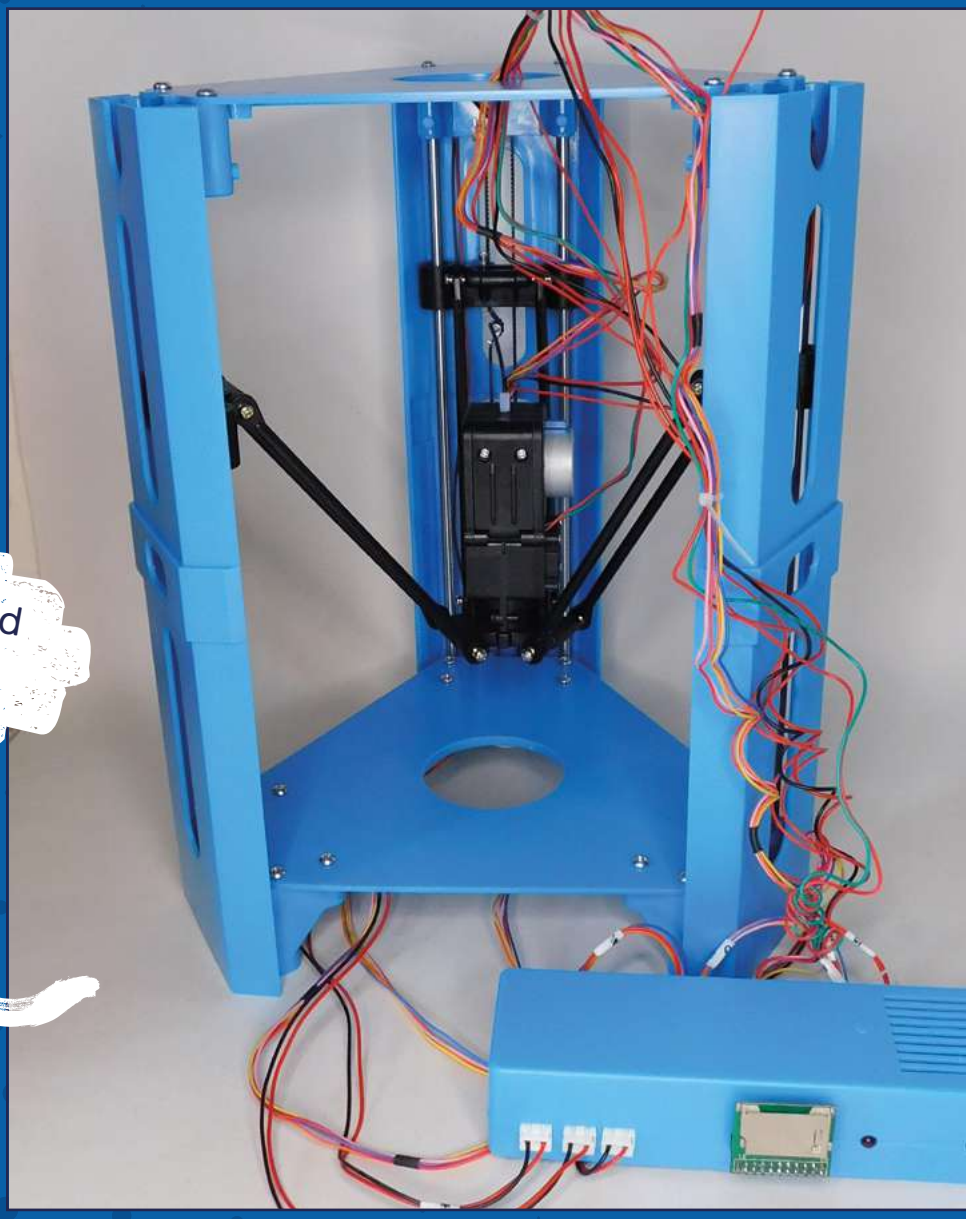
We have two concerns about this printer. Firstly, there's the print volume, which is just 10×10×10cm. This is big enough if you want to print figurines and other small objects, but too small for many more functional prints. Secondly, we're not sure how long we would expect the small stepper motors to last. When we've used these little drivers before, we've found them prone to wearing out and breaking quite quickly. This machine survived our testing, but we're not sure how much longer it would last.

Neither of these concerns is a deal-breaker considering how easy this machine was to put together, the print quality, and the price. For many people, these would be acceptable trade-offs for a first 3D printer. →

"THIS FRAME ISN'T THE MOST ROBUST"

10 HERO

PRINTING IN A DIFFERENT DIMENSION



We paid £85

"THE RESULT IS HYPNOTIC TO WATCH"

This 3D printer launched to much fanfare as the first 3D printer under $100. In fact, we looked at it in the very first issue of HackSpace magazine. It's still selling for around its original price, so we decided to see if the current version was the same as the initial version, and how it compared to the other printers that have now entered this price bracket.

Unlike the other printers on test, this is a delta printer. Rather than having three motors that run at 90 degrees to each other (in the X, Y, and Z axes), this has three motors that each move drive-belts vertically. The print head hangs from these drive-belts and, with a bit of clever mathematics, it can be positioned just as a traditional X, Y, Z printer can. The result is hypnotic to watch.

The 101Hero is straightforward to put together. The three pillars are attached to identical top and bottom plates. A few screws hold everything together – it's perfectly possible to be ready to print within an hour.

The first sign that things aren't entirely user-friendly came when we tried to level the bed. In fact, this isn't really levelling the bed (since the bed is fixed), but adjusting the stop heights of the three motors. Using these, we can tell the printer to start slightly higher or lower, and so ensure that the print head is the right distance from the print bed. However, there's no firmware support to make this easy. You have to start a print, see how it goes, make some adjustments, and restart it. There's not even firmware support for moving the print head. You have to start the print, see how it's going, pull the plug to stop the print, then with the power off, yank the motors up because you can't lift the print head off the bed with the control unit (there is only a single button to reset it), take the failed print off, and start again.

Similarly, there's no support for loading filament. You have to take the front off the extruder, push it in as far as it can go and hope that enough is extruded in the brim for the filament to make its way through before the main print starts.

We had a few OK prints from this printer – not many, but a few. And some catastrophes. There's a small community around this printer who have shared some modifications that you can make to get something resembling acceptable prints out of it, but out of the box, it's iffy at best.

There's very little to recommend this printer. It's annoying to use, and the print quality varies from questionable to non-existent. Perhaps it would be acceptable if it were the only option in this price bracket, but it isn't any longer, and we had far better results from others on test. →

FMEA

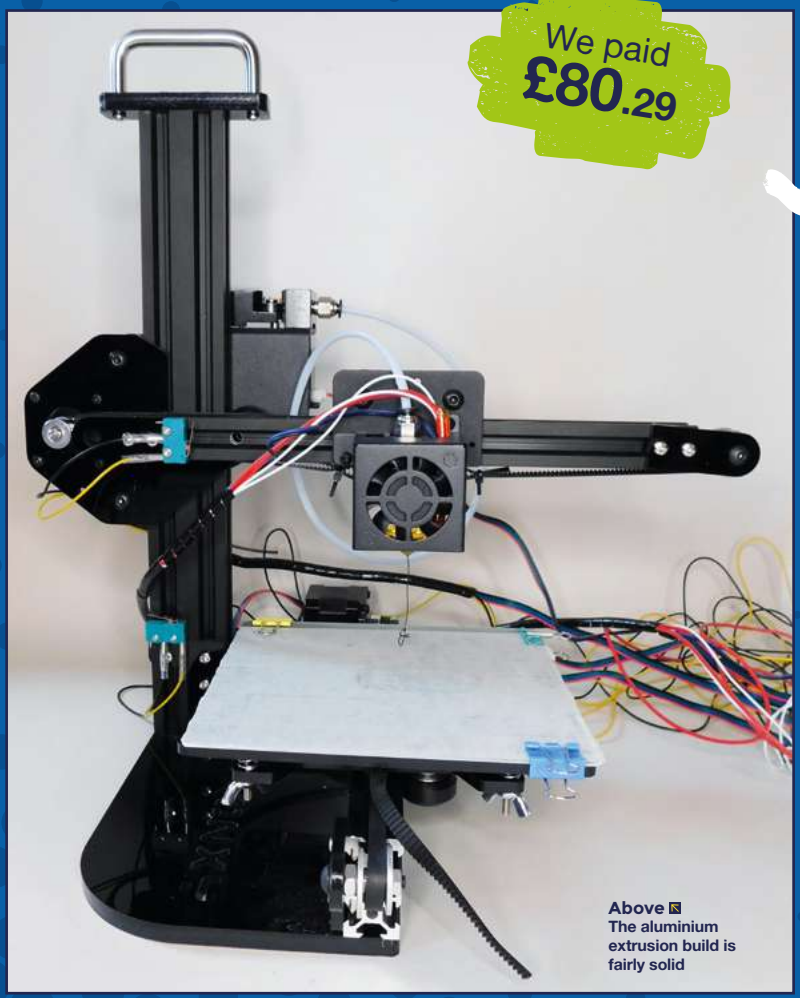
We were intending to test five printers out in this feature, but one arrived without building instructions, and we weren't able to find any online. The box contained a large array of aluminium extrusion and laser-cut parts. After a valiant attempt, we've been unable to get this printer assembled in time for this feature. If we can work out how to put it together, we'll look at it in a future issue.

Below
Some people claim to be able to get reliably OK prints off the 101Hero, but we couldn't

TRONXY X1

A LITTLE PRINTER WITH BIG AMBITIONS

We paid £80.29

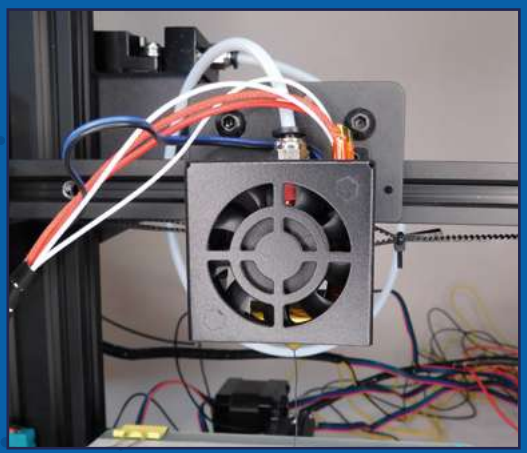


Above
The aluminium extrusion build is fairly solid

This printer comes in a lot of very small parts, and building it is quite an involved process. It should be possible in a day, but we wouldn't recommend scheduling anything else on that day. Fortunately, the instructions (that came on an included microSD card) were detailed and made it a reasonably straightforward (if not quick) build.

Once fully assembled, you get a 15×15×15cm build volume. While instinctively, this may not sound much bigger than 10×10×10cm (on the EasyThreed X1), but it's actually over three times the size, so while this is limiting, it's not *as* limiting.

Below
It took a bit of work to get the belt tension right, but when we did, it printed well



Levelling the bed proved a bit of a challenge because the entire print bed rocked side to side about 5 mm (at the edge). This didn't actually cause us too many problems in printing as it was just about stiff enough to not rock when in use. It didn't become any worse during our testing, but it may be a larger problem over time (Thingiverse user PB2000 has created a printable modification to improve the situation: **hsmag.cc/TronxyBed**).

Our first couple of prints suffered badly from loose belts, but after a bit of levering and pulling, we tightened those up and started to get more reliable prints.

There's no part cooling fan, but in our testing, we were able to get reasonable results even on overhangs and bridges. Perhaps this is due to the lack of heated bed helping everything cool down a bit quicker. It should be straightforward enough to add a part cooling fan later, should you wish.

Without a heated bed, you're more or less limited to just PLA filament (more exotic filaments usually require a heated bed to help them stick and avoid warping), but for many people, that's perfectly acceptable.

We were impressed by this printer. The aluminium extrusion frame and chunky stepper motors felt robust enough for regular use. With the bed-stabiliser modification linked above, you should have a more reliable printer, and although the print volume isn't huge, it is big enough for a range of prints. The frame proved stiff enough for accurate prints, and although it didn't quite manage the detail of some more expensive printers we've tested, it did print reliably to a quality we were happy with. →

"Without a heated bed, you're limited to PLA filament"

Print volumes

How much space do you need to print? There isn't an easy answer to this, and it depends on what you want to print. If you want to print figures and miniatures, then a small print volume isn't usually a problem. If you want to print cases for your projects, then it depends a lot on the size of the projects you want to encase.

In the HackSpace lab, we have a range of printers with different print volumes, and we still find that there are things we want to print larger than any print volume available to us. Ambition will always exceed technical capability!

Volume is usually a trade-off against price as larger print volumes need not just larger frames, but more robust frames as well. Printing large objects is also incredibly slow. Filling even a modestly sized 20×20×20 cm build volume can easily take several days to print, depending on your settings.

Remember that you can always divide a larger object into multiple parts and glue them together after they've printed.

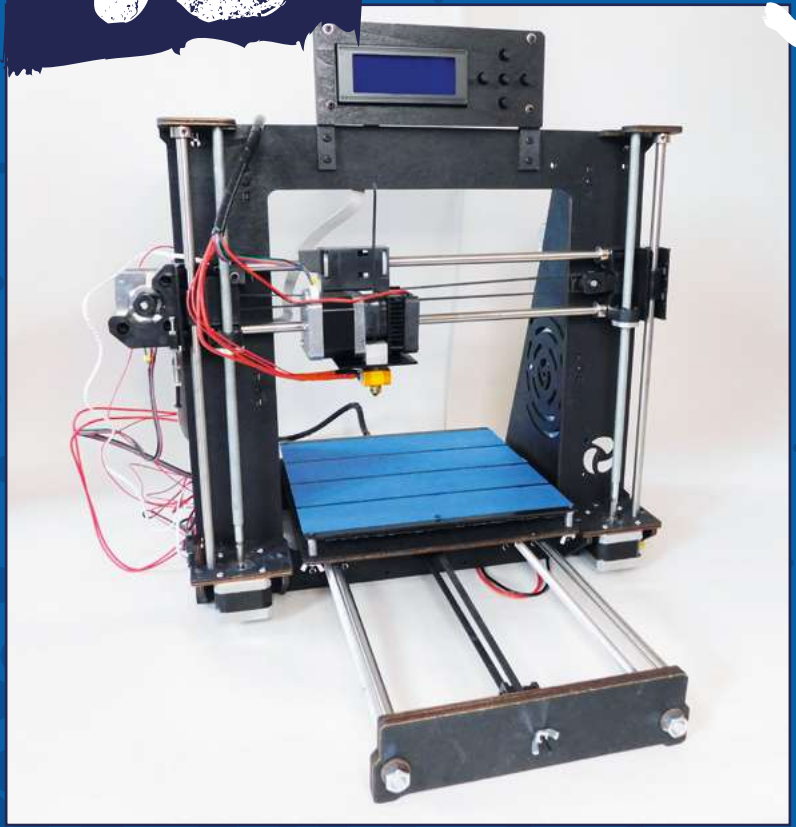
Above
The separate control box gives a powerful interface for controlling the printer

Above
With minimal tweaking, we were able to get working prints out of this machine

CTC PRUSA i3

REMAKING A CLASSIC

We paid £71.88



This printer is made mostly from laser-cut plywood that slots together and is secured with M3 nuts and captive bolts. You get considerably more printer than in the other ones on test: it has a larger print volume (20×20×16 cm), and this is supported by a full frame with dual Z-axis motors for increased stability. As the name suggests, this is based on the open-source Prusa i3 design that is now very familiar to many people in the 3D printing world.

The printer comes partially assembled, so it was pretty straightforward to put together – it took about an hour in total. However, we did have a few issues with the build. The metal-cased power supply required us to open it (breaking the 'warranty void' sticker) to see if it was set to 240 V or 120 V. Ours was set correctly, but it wasn't a particularly comforting experience for people not used to working with the internals of high-voltage systems (see more on this in the Safety box, opposite).

Another problem with this printer is that the end-stop for the Z-axis was set so that the printer head hit the bed even with the bed lowered as far as it could go. We solved this by joining two bolts together and using them to trigger the Z-stop switch with the extruder at a higher position.

This printer came with a heated bed, but that bed wasn't particularly flat, and it dipped several millimetres in the middle. It's a fairly simple process to clip a glass bed to it, and you should be able to get one and still keep this printer under £100.

SAFETY

While this printer runs with its initial setup, we can't comfortably say that we'd recommend running it as it is. The high-voltage side isn't sufficiently well-guarded, and it wouldn't be too much of a stretch to imagine a finger accidentally coming into contact with 240 V. If you're suitably experienced with high-voltage electronics, you could build a case or some other mounting to protect the mains voltage from any bits of flesh. Make sure that there's sufficient ventilation to keep the electronics cool. This is the only printer on test that didn't use a sealed power supply and involved wiring the high-voltage connections. This is also the only printer on test with a heated bed, which is another potential source of safety problems.

The heated bed makes the problem of part cooling more serious, as the previous layer doesn't have as much time to cool before the next layer is printed on top of this. It would be a simple process to add a part cooling fan. Alternatively, you could point a desk fan at the printer. It's a little less precise, but does improve things significantly.

Print bed aside, this printer worked with a basic setup. CTC doesn't have a recommended slicer setup, so we started with a standard Prusa i3 profile and went from there. Slowing it down a little gave the plastic more time to cool before the next layer was deposited on top. It helped a little, but really, this needs a cooling fan if you want to print overhangs or larger parts.

There is a bit more wobble in this printer than we'd like. If you look at the prints, there's inconsistency in the surface finish. It's small enough to be cosmetic for most purposes (but could cause problems if you're working with small tolerances). You may be able to get rid of some of this by tuning the slicer settings, but fundamentally, the problem is in the frame. The other printers in this test mitigate this by their small size, but with a larger print volume, this printer is more prone to wobble. You may be able to make some modifications to this printer to reduce the wobble, but really, it's a pay-off for having a larger format printer at a low price.

We got some OK prints from this printer. The combination of wobble and sagging (due to lack of part cooling) didn't leave a particularly good finish. We suspect that with some work on both stiffening the physical structure of the frame and tuning the slicer profile, this quality could be improved somewhat. Fundamentally, though, the size and materials of the frame are working against you.

Compared to the other printers on test, this is far more feature-packed. The heated bed makes prints stick more reliably, and gives the option for printing with a wider range of materials. There are natural upgrades to a glass bed and part cooling fan if you want to improve the quality. However, packing in all these features, while keeping the price down, means you have to compromise more on print quality than with the other printers in the price bracket. You also have to keep in mind the issue with high-voltage electronics (see Safety box). Whether or not that's a good compromise depends on how you want to use your 3D printer, and how long you want to spend tinkering with it. →

Above
Too much wobble and stringing for our liking

Below
The high-voltage electronics are exposed on the side of the printer

Right
Wobble means that each layer isn't lined up with the one below it

VERDICT

When we started this feature, we were genuinely unsure if you could get a working 3D printer for under £100. Many printers that are usually considered budget cost twice the price, so perhaps it was just a pipe dream that any of these machines would be able to produce anything with hot plastic. We were pleasantly surprised.

Obviously, in this price bracket, you have to be prepared to make compromises, but fortunately, you do have a choice of what compromises to make. None of these printers was capable of really high-detailed prints, but two were capable of making prints 'good enough' for many use cases. Weirdly, these printers were both called X1.

The EasyThreed X1 was wonderfully easy to put together and printed really well out of the box. However, it's small, slow, and may not last as long as some of the others.

"YOU WILL HAVE TO MAKE COMPROMISES"

The Tronxy X1 is the most complete printer. It's larger than the EasyThreed X1 and printed well (once we'd properly tightened the belts). A modification to stop the bed wobbling is recommended; this should be pretty straightforward to add. A part cooling fan will improve your prints, but without a heated bed, it's not essential.

We also want to give a mention to the CTC Prusa i3, which is the most feature-packed printer on test. However, it needs some work to modify the electronics housing, and the bed needs an addition to make it flat enough. Ideally, it needs a part cooling fan, and it would benefit from some work on the frame. This isn't so much a finished printer as a starting point for people who enjoy tinkering with printers. For this price, though, it could be an option.

We can't see a situation in which we'd recommend the 101Hero. There is a community of enthusiasts around the printer, and some people seem to have been able to get OK quality prints off it. However, it doesn't offer anything that other printers in the same price bracket can't do better. ◘

SPENDING MORE?

While we did get three working printers for under £100, we shouldn't forget that this is the lowest end of the market. If you have a little more money to spend, you have more options.

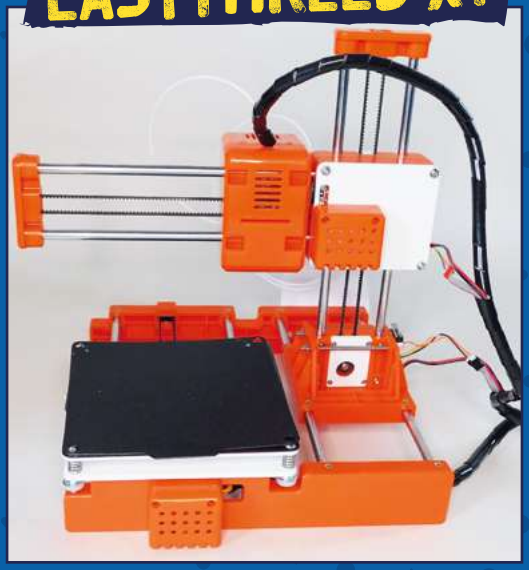
At around £200, the Creality Ender-3 has come to define budget 3D printing. It's built of extruded aluminium, which gives a sturdy frame that should last. One of the best features is the community of enthusiasts and ecosystem of upgrades that have sprung up around this printer, which means it's a printer that you should be able to keep running, and that can grow with you as you want more out of your 3D printer.

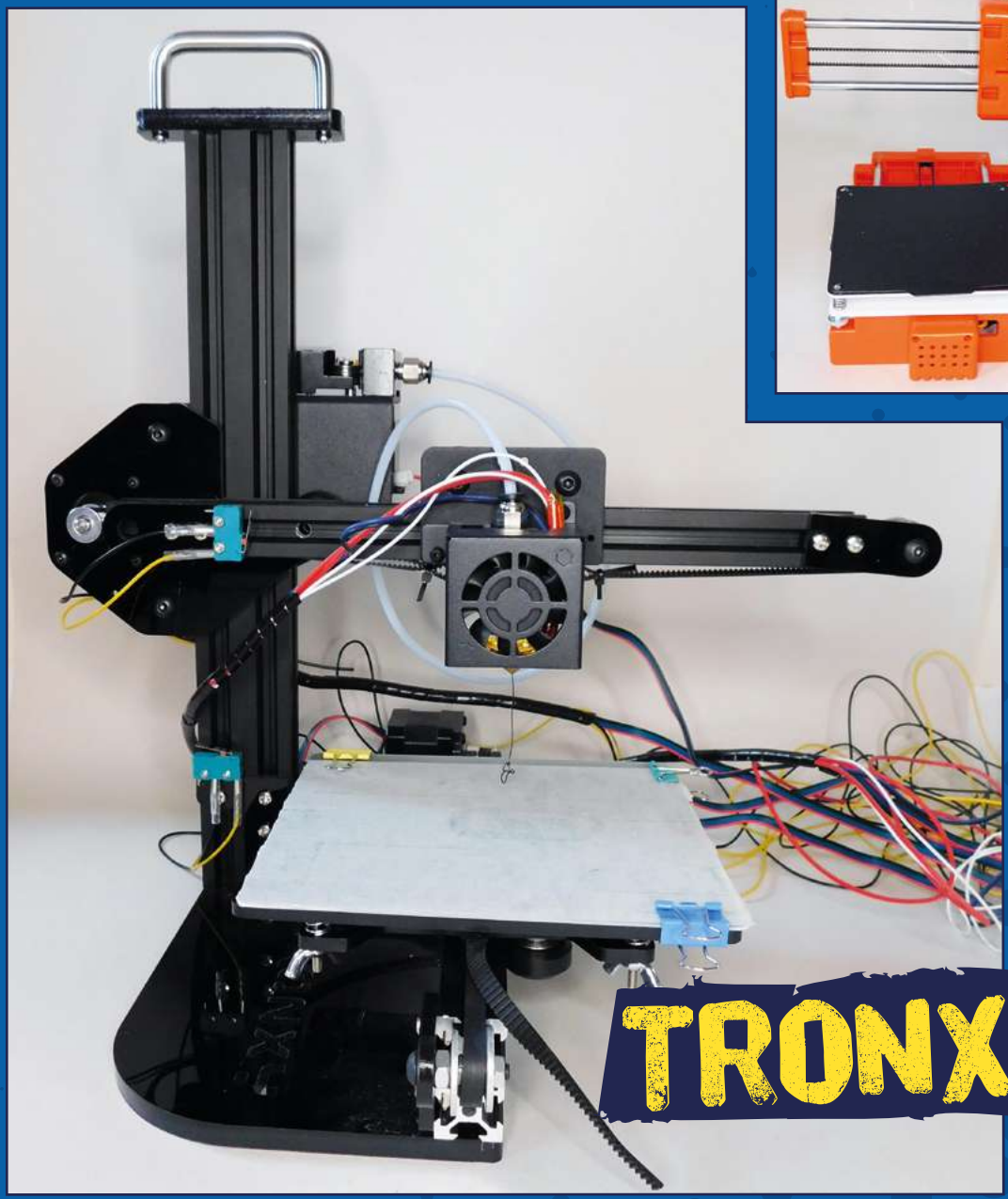
At around £300, you can get an Original Prusa MINI. While no 3D printer is guaranteed to work 100% of the time, Prusas (from the Prusa company, as opposed to open-source Prusa designs such as the CTC Prusa i3) have become well-known for 'just working'. Along with a solid printer, they offer good support and can provide a fuss-free way into 3D printing.

These are just a few examples that we've tested out. There's a wide range of 3D printers at many different price points. There's no such thing as a perfect 3D printer as they all have compromises – only you can decide what's right for you.

BEST PRINTERS UNDER £100

EASYTHREED X1





TRONXY X1

FEATURE

How I

BIT-BANGED DVI ON THE RP2040 MICROCONTROLLER

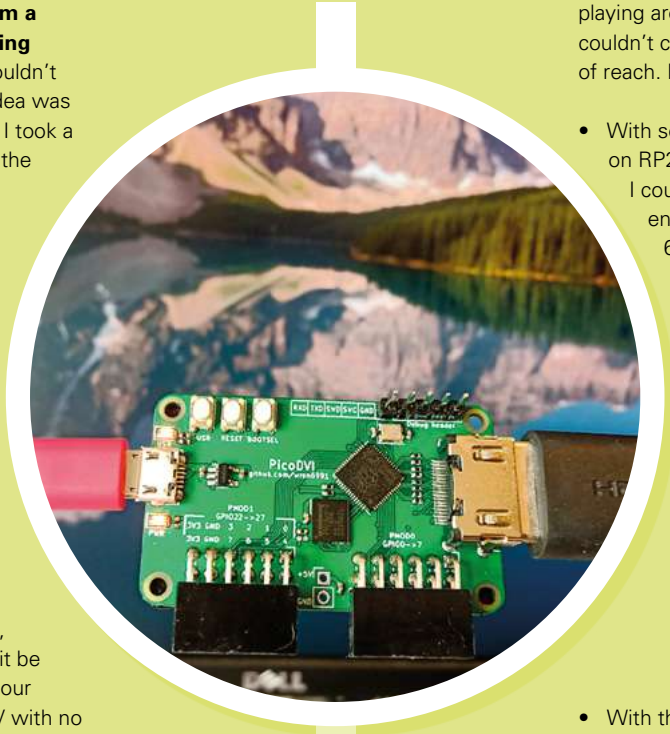
Get digital video out of a microcontroller

By **Luke Wren**

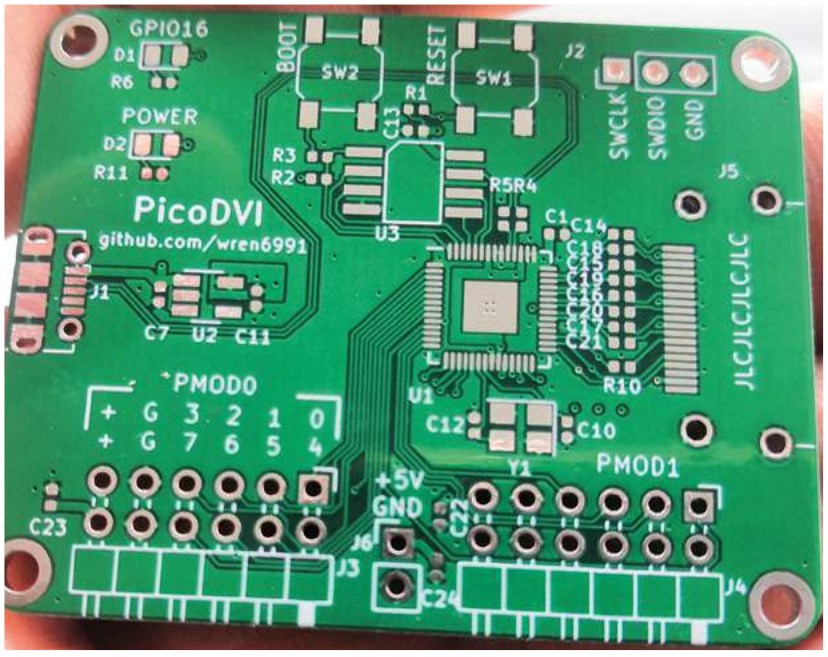
This project stems from a stupid idea I had during RP2040 bring-up. I couldn't convince myself the idea was too stupid to work, so I took a leap of faith on it, and the results are documented here.

RP2040 was designed to run at 133MHz, but we found (without too much surprise) that typical silicon can be pushed further. In fact, there was an overlap between the maximum system clock and the TMDS (transition-minimised differential signalling – a method for transmitting high-speed digital data) bit clocks of slower DVI (digital video interface) video modes.

We had done great stuff with VGA on the FPGA platform (before the Pico took its final form), which ran at 48MHz, but wouldn't it be absurd and wonderful to connect your microcontroller straight to an HDTV with no other electronics in between? This seemed unlikely to work out, but I stayed up at night playing around with assembly loops, and I couldn't convince myself that DVI was out of reach. Everything seemed to fit:



- With some of the core-local hardware on RP2040, and a neat encoding trick, I could do pixel-doubled TMDS encoding on the fly using around 60% of an ARM Cortex-M0+ processor (running at 252MHz, for 640×480p 60Hz DVI)
- PIO can yeet out data streams at system clock frequency, and drive a 1/10th rate clock on the side, with pretty minimal programming
- Some of the DMA features help with putting together the sync/blanking patterns on the fly, rather than having the patterns flat in memory
- With the second processor utterly unencumbered, you can render some pretty graphics to put on your DVI display



Left
The bare PCB is ready for soldering

The greatest unknown was driving 252Mbps serial through the general-purpose digital pads (especially *differential* serial, emulated with two single-ended pads). By this point, I was utterly driven and consumed by the need to find out if DVI could work, so I laid out a board over a few evenings after work.

The Rev A board uses a slightly cursed coupling circuit, which I first saw (and used) on the ULX3S FPGA board; it connects 3V3 I/Os straight into the HDMI socket through some coupling caps.

Those who understand the TMDS physical layer are probably screaming. I was fine, though, because I did not read the electrical section of the spec until after I got this board working. Then I screamed. Before the boards arrived, I did some debugging with these two strategies:

- Run the entire system at 12MHz (crystal frequency) so that the signals are probeable, but the relative speed of I/O, DMA, and CPUs is the same. This makes sure my code can keep the PIO state machines fed with data.
- Swap in an alternate PIO program that outputs 10-bit UART data frames instead of direct serial (a 17% drop in throughput). I could then dump the TMDS stream with a logic analyser, and examine and parse it on my machine.

Those who understand the TMDS physical layer are probably screaming. I was fine, though

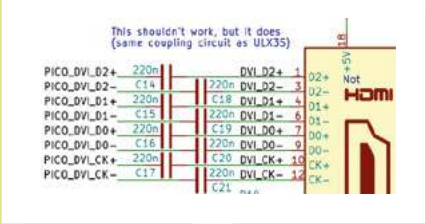
I also tried out my slightly hare-brained TMDS encoding scheme – which matches the letter but not the spirit of the DVI specification – on an FPGA board with some DVI gateware that I wrote for a previous weekend project. This confirmed that the principle was sound, and that my TV and monitor would have no trouble with the output of the matching software encoder on RP2040, provided the chip could physically shove bits out of the pins fast enough.

Because this is a home project, I didn't touch the HDL simulator, and stuck to ARM debugger, UART, and logic analyser for my debugging. This worked some kinks out of the software, and bring-up of the freshly soldered board was smooth. After swapping the blue and red lanes into the right order – to which I will say, in my defence, I consistently thought the blue+sync lane was lane 2 – I had a clean RGB565 QVGA 60Hz static image on my monitor.

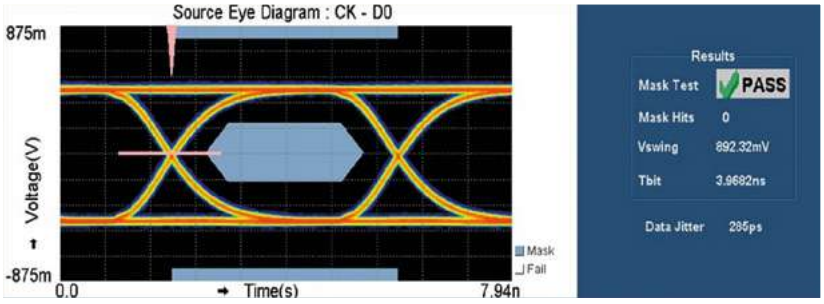
IMPROVED OUTPUT CIRCUIT

After reading the TMDS electrical section of the DVI spec – and staring quietly out of the window for a while, wondering how this board *ever* worked – I rethought the output circuit. Eight capacitors was clearly not the way to go – what I really needed was eight *resistors*. That's what I call a DVI PHY.

Below
Make sure you get the right passives!



FEATURE



Index	Test Name	Lanes	Spec Range	Meas Value	Result
1	7-9 : Source Clock Jitter	CK	Clock Jitter < 0.25*Tbit;	0.057*Tbit	Pass
2	7-10 : Source Eye Diagram	CK - D0	Data Jitter < 0.3*Tbit;	0.07*Tbit	Pass
3	7-4 : Source Rise Time	CK	75.00ps < TRISE;	1.0882ns	Pass
4	7-4 : Source Rise Time	D0	75.00ps < TRISE;	1.0924ns	Pass
5	7-4 : Source Fall Time	CK	75.00ps < TFALL;	1.1128ns	Pass
6	7-4 : Source Fall Time	D0	75.00ps < TFALL;	1.1304ns	Pass
7	7-8 : Max Duty Cycle	CK	Max Duty Cycle < 60.0%;	50.65%	Pass
8	7-8 : Min Duty Cycle	CK	40.0% < Min Duty Cycle;	49.39%	Pass

Right
PicoDVI produces valid HDMI signals

I also revised my earlier approach of 'turn all the GPIOs up to 11', and reduced the pad drive and slew. At work, a colleague agreed it would be a great idea to plug my microcontroller monstrosity into the scope setup we use for 4K HDMI testing. Here are the results at VGA 60Hz (252Mbps).

I was sitting on the other side of the lab while he was running the test, and when the eye mask appeared, he just said, "Do you wanna see something funny?"

This is a silly amount of bandwidth for a tiny little microcontroller

A clean bill of health! We also tried 720p30 (372Mbps), which requires overvoltage on typical silicon (something you can do with one register write on RP2040).

Honestly, this has shaken me. This is a silly amount of bandwidth for a tiny little microcontroller.

Although it passes the eye mask test and a few other tests, this circuit is not fully compliant with the DVI spec. In particular, our logic '1' is not quite right, due to the CMOS drive on the GPIOs: any more than a ~60mV mismatch between the source and sink +3V3 rails will push our high-level offset outside of the +- 10mV allowed by the spec. This is a real nitpick, because a *differential* receiver is unlikely to care about a 10mV *common-mode* offset, but still, it is out of spec. A better circuit could use a fast silicon diode and a smaller resistor value, for example, 220ohms, so that the emulated CML output floats on the sink's +3V3 supply when we output our 3V3 CMOS high level, but still sinks the requisite 10mA when driving low.

That said, it's compliant enough that I can wander around the office and plug it into every monitor I see without any of them exploding (if my manager is reading this – hello).

GOING FURTHER

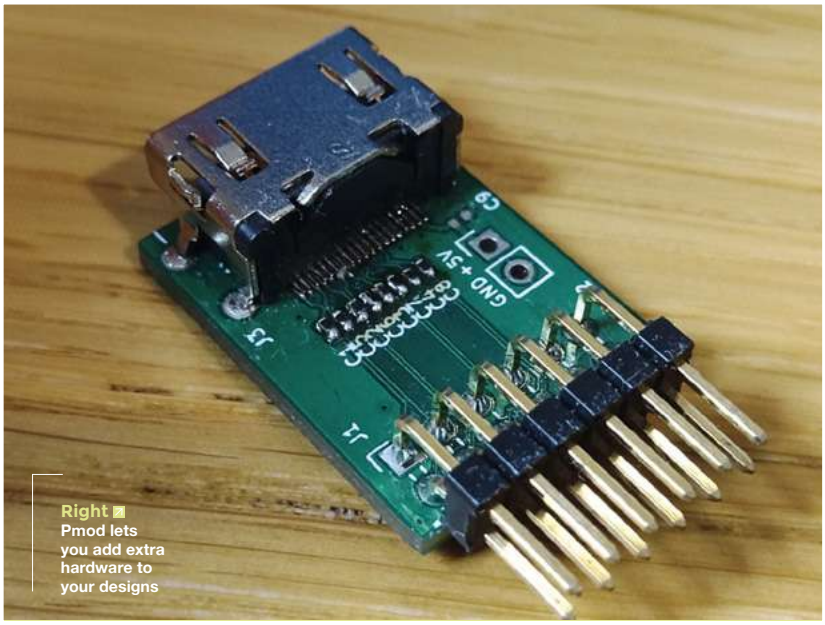
Everything we have done is software-defined – there is no video hardware on this chip. That would of course be silly on a microcontroller. Let's list all the hardware resources used to display a pixel-doubled image on screen:

Three out of eight PIO state machines (the DVI code requires these all be on the same PIO instance, of which there are two, with four state machines each)

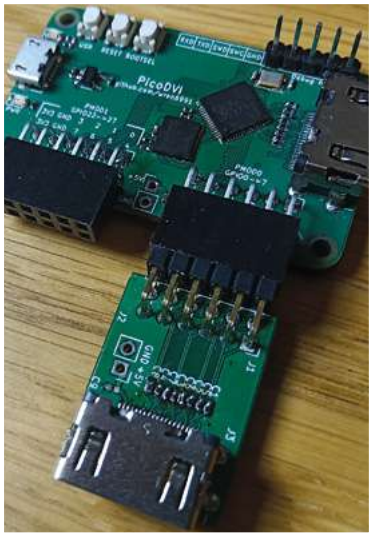
- Six out of twelve DMA channels (two per TMDS lane: one for control blocks, one for data)
- 30% of DMA bandwidth and PIO bus endpoint bandwidth
- 60% of CPU cycles on one core, other core 100% free
- Just over 50% of RAM with a QVGA RGB565 image (but RGB332 support is simple enough)
- The PicoDVI board's only HDMI-shaped socket

Hmm. All of these numbers are less than half of the total, and everything else is software. It's a shame there's only one socket I can put an HDMI cable in. I mean, I guess I do have these adorable Pmod DVI adapters that I keep plugging into FPGA boards and getting away with it:

Oh. Maybe? It fits…



Right
Pmod lets you add extra hardware to your designs



Above
When one HDMI just isn't enough

I guess the jig is up at this point, because of course, I wouldn't post something so daft-looking if it didn't work.

The code is here: **hsmag.cc/PicoDVIcode**, and there are a few examples of this mind-bending technology in use up on my GitHub page: **hsmag.cc/PicoDVIexamples**.

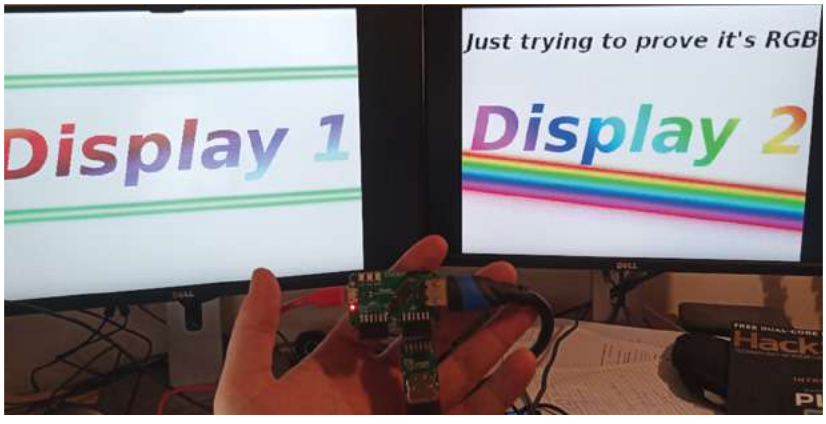
ENCODING TMDS

DVI uses an encoding scheme called TMDS during the video periods. Eight data bits are represented by a 10-bit TMDS symbol, which is serialised at 10× the pixel clock. Three lanes transfer 24 bits of data per pixel clock, which for our purposes is one pixel. TMDS is DC-balanced, although DVI as a whole is *not* DC-balanced on all lanes due to the control symbol encoding. The algorithm given in the DVI spec is quite fussy, and you are supposed to match its output exactly. It tracks running disparity with a counter, and optionally inverts symbols to bound the disparity, with some tie-break rules for 0-balance symbols.

Here's a key fact about TMDS: if the current running disparity is 0, and you encode data `x` followed by data `x ^ 0x01`, this produces a pair of TMDS symbols with 0 net parity. If you manipulate the input data in this way – duplicating the pixels, and twiddling the LSB – TMDS becomes *stateless*, because the running disparity is defined to be 0 at the start of each video period, and returns to 0 after each duplicated pixel pair.

If we have a half-resolution scanline buffer, and are only interested in seven or fewer bits of significance for each colour channel, we can encode this with a lookup table (LUT), where each entry is two TMDS symbols with net balance 0. The toggling of the LSB with each output pixel is not noticeable.

Great. LUTs are fast. On an ARM Cortex-M0+, though, they're not *that* fast. Each load/store is two cycles, and we end up spending a surprising amount of time shifting and masking the data. Here's a vaguely plausible loop for encoding →



Right
Two HDMI displays running on one RP2040

Left
Of course, there has to be Bad Apple

one colour channel, compiled with `-O3 -mcpu=cortex-m0plus` on gcc 9 (**hsmag.cc/GodBolt**).

```
*C*

```
void tmds_encode_16bpp(const uint16_t
*pixbuf, uint32_t *tmdsbuf, size_t n_pix,
uint16_t chan_mask, unsigned int chan_
shift) {
    for (size_t i = 0; i < n_pix; ++i) {
        unsigned int idx = (pixbuf[i] >>
chan_shift) & chan_mask;
        tmdsbuf[2 * i] = tmds_table[idx];
        tmdsbuf[2 * i + 1] = tmds_
table[idx + 1];
    }
}
```

ARMv6M

```
tmds_encode_16bpp(unsigned short const*,
unsigned long*, unsigned int, unsigned
short, unsigned int):
        push    {r4, r5, r6, r7, lr}
        ldr     r7, [sp, #20]
        cmp     r2, #0
        beq     .L1
        lsls    r2, r2, #3
        ldr     r5, .L6
        adds    r2, r1, r2

.L4:
        ldrh    r4, [r0]     ; 2 cyc
        adds    r0, r0, #2   ; 1 cyc
        asrs    r4, r4, r7   ; 1 cyc
        ands    r4, r3       ; 1 cyc

        lsls    r6, r4, #2   ; 1 cyc
        ldr     r6, [r5, r6] ; 2 cyc
        adds    r4, r4, #1   ; 1 cyc
        str     r6, [r1]     ; 2 cyc

        lsls    r4, r4, #2   ; 1 cyc
        ldr     r4, [r5, r4] ; 2 cyc
        str     r4, [r1, #4] ; 2 cyc

        adds    r1, r1, #8   ; 1 cyc
        cmp     r2, r1       ; 1 cyc
        bne     .L4          ; 2 cyc if
taken
.L1:
        pop     {r4, r5, r6, r7, pc}
```

Focusing on the loop starting at `.L4`, this is a surprisingly literal translation – first, load a pixel, bump the pointer, mask, and shift (five cycles). Next, transfer a pixel from the LUT to the output buffer (five cycles) while bumping the LUT index (one cycle), transfer the second pixel (five cycles), do a little bookkeeping, and branch back to the start (four cycles). Each loop takes 20 cycles and encodes one colour channel of two output pixels.

We must output a pixel once per ten system clock cycles (as the system runs at the TMDS bit clock), and since we are doubling pixels horizontally, we may as well double vertically too, by using each encoded buffer twice. Taking horizontal blanking into account (1:4 ratio at VGA), we would spend 1.2 of our two cores on TMDS encode, and have 0.8 cores left to generate DVI timing and render graphics.

The compiler has messed up here (or perhaps I have unwittingly constrained it to produce bad code by writing shoddy C), and we can save four cycles right off the bat with better instruction selection:

```
.L4:
        ldrh    r4, [r0]     ; 2 cyc
        adds    r0, r0, #2   ; 1 cyc
        asrs    r4, r4, r7   ; 1 cyc
        ands    r4, r3       ; 1 cyc

        lsls    r4, r4, #2   ; 1 cyc
        ldr     r6, [r5, r4] ; 2 cyc
        stmia   r1!, {r6}    ; 2 cyc

        adds    r4, r4, #4   ; 1 cyc
        ldr     r6, [r5, r4] ; 2 cyc
        stmia   r1!, {r6}    ; 2 cyc

        cmp     r2, r1       ; 1 cyc
        bne     .L4          ; 2 cyc if
taken
```

But this is still painfully slow – it wouldn't even fit on one core.
```

Here are some avenues for improvement:

- Use a word load to fetch two input pixels at once, so we can amortise the load cost, and some of the shift/mask cost
- Use larger **ldmia** and **stmia** on the LUT to squeeze more memory bandwidth out of the M0+ (**ldr** is two cycles and **ldmia** is n + 1)
- Use the interpolators on RP2040 to accelerate address generation

The interpolator is a fun piece of hardware for accelerating fixed-point arithmetic. Note: 'interpolator' is a working title which we will definitely not forget to change to something better before launch. The original plan was a simple, configurable 2D phase accumulator that we could use to play Super Mario Kart on the FPGA platform at 48MHz. Sadly, that game port never materialised, not least because we couldn't publish it. After a lot of back and forth between hardware and software, we realised that making the data path a little more flexible would go a long way, and this eventually led to the current guise of the interpolator.

Our trick here is loading a one-word pixel pair into one of the accumulators, and configuring the interpolator to extract the correct bits of each pixel, shift them, and add them to a LUT base pointer.

The interpolator doesn't have a left shift (not needed for Super Mario Kart), so for the blue channel (least significant in our RGB565 pixel format), we need to do one left shift per two pixels on the processor, to scale up to the LUT entry size. For the other channels, we can use a different loop, without the left shift. Here is the encode loop from **tmds_encode.S**:

```
// r0: Input buffer (word-aligned)
// r1: Output buffer (word-aligned)
// r2: Input size (pixels)
// r3: Left shift amount

decl_func tmds_encode_loop_16bpp_
leftshift
        push {r4, r5, r6, r7, lr}
        lsls r2, #3
        add r2, r1
        mov ip, r2
        ldr r2, =(SIO_BASE + SIO_
INTERP0_ACCUM0_OFFSET)
        b 2f
.align 2
1:
```

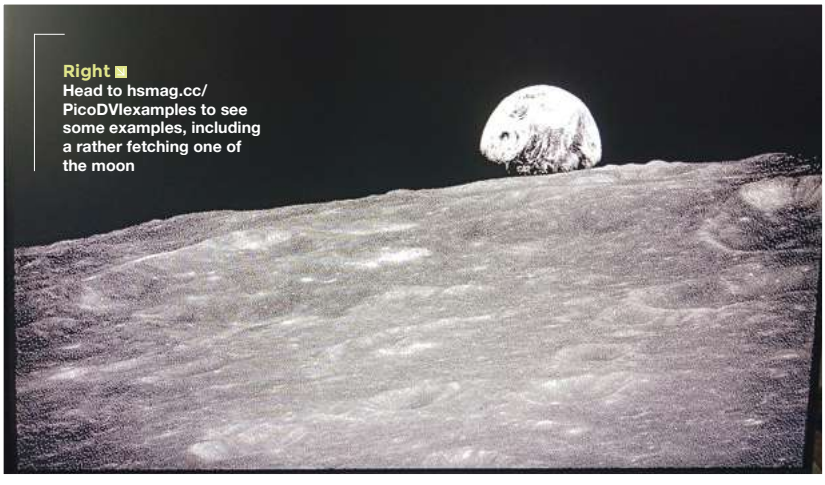
```
.rept TMDS_ENCODE_UNROLL
        ldmia r0!, {r4}             ; 2
cyc
        lsls r4, r3                 ; 1
cyc
        str r4, [r2, #ACCUM0_OFFS]  ; 1
cyc
        ldr r4, [r2, #PEEK0_OFFS]   ; 1
cyc
        ldmia r4, {r4, r5}          ; 3
cyc
        ldr r6, [r2, #PEEK1_OFFS]   ; 1
cyc
        ldmia r6, {r6, r7}          ; 3
cyc
        stmia r1!, {r4, r5, r6, r7} ; 5
cyc
.endr
2:
        cmp r1, ip                  ; 1
cyc
        bne 1b                      ; 2
cyc if taken
        pop {r4, r5, r6, r7, pc}
```

Assuming **TMDS_ENCODE_UNROLL=1**, each loop iteration takes 20 cycles, and produces four output pixels. Taking the lack of left shift for red and green into account, this works out to 58% of a core to do TMDS encode, which is just over 2× faster than the compiled LUT loop. We can get another 10–15% performance by increasing **TMDS_ENCODE_UNROLL**.

> The original plan was a simple, configurable 2D phase accumulator that we could use to play Super Mario Kart on the FPGA platform

We handle 8bpp pixels in a similar way, but use both interpolators – one to extract pixels 0 and 1 from the loaded word, and one for pixels 2 and 3. 8bpp encode is slightly faster than 16bpp encode, because we get four pixels for each load from the pixel buffer.

Getting DVI out of RP2040 is possible, but it requires pushing the little device to it's limits. The code is all on Github, so take a look and see if you can push it any further!



Right
Head to hsmag.cc/PicoDVIexamples to see some examples, including a rather fetching one of the moon

HackSpace magazine meets...

Geeky Faye Art

What would you do if you could do what you want?

Active members of the online maker community may already be familiar with Geeky Faye Art, the London-based, US-born maker with a passion for illustration, animation, and 3D printing. For everyone else, Alex Bate took the time to (virtually) sit down with Faye to learn more about what she does and why she can't get enough of 3D printing, accessibility, and expanding her creative skill set.

Faye has always been artistic, having studied Media Arts and Animation at university before pursuing a career in digital art for various well-known brands. But how does one go from making animations for utility companies to filming 3D-printing video tutorials on YouTube, and releasing their first product, a fully customisable, 3D-printable picture frame? →

Right
What happens when you mix PLA and alcohol markers? find out at hsmag.cc/2WVprl

INTERVIEW

"I started drawing when I was a kid," Faye explains when talking of her career. "I picked up a pencil and a piece of paper and pretty much never put it down." So, it was no surprise that, when it came to making university choices, she decided she wanted to study art and picked the course she felt would have good job prospects on the other side.

"Digital was the future, and I was able to see how broadly 3D art could be applied as it is used in so many industries. It's a weird thing to do professionally because you're not necessarily tied to a specific industry – you have a position that is potentially hireable in a range of industries, and I have worked in many of them."

Jumping on an opportunity to move to London from the US ten years ago, something she'd wanted to do since her childhood, Faye began working for a variety of organisations, both in full-time positions and freelance. But it wasn't always smooth sailing.

"I've had some bad luck. I've had jobs pulled out from underneath me, and have worked for companies that either went under or were bought out – it's the same outcome either way. I've had some really bad luck." And the bad luck began to wear on her. "I was doing this for a long time. I was good at it. And I hated it. I wasn't very happy." And all this bad luck and unhappiness eventually brought Faye to the point of complaining to a friend that she wanted to change her career.

"I spent some time trying other stuff and not being happy. It was a friend of mine who asked, 'What do you want to do? Really, if you could do anything, and money didn't matter, what would you do?' and I said I'd make stuff. This is what I love to do. But not just one thing, everything. I'm a bit of a generalist. I've never been able to specialise in one thing.

BACK TO MAKING

"And, because I'd spent a lot of years ignoring my traditional skills in favour of digital, I was hungry to do something with my hands again. I wanted to make stuff, and do more cosplay, because I've always loved cosplay as it's basically making, and theatre, and being a nerd all pushed into one.

"I'd make what I want to make. I wouldn't make for the purpose of selling a product or making a company money. I would do the ideas that I wanted to do, and I would do them for me. That's what I would do, I told her." And when her friend replied, 'OK, do that,' Faye failed to find an excuse not to. "I've always been good with money, and I'd managed to squirrel some away when I was working freelance, so I could take a little risk, and take a bit of a chance."

Such a career move may seem impossible to many, but with the introduction of membership platforms like Patreon, and the ability to gain revenue via Google AdSense and sponsorship, this dream is closer to becoming a reality than it ever has been before. It's something a lot of older makers new to the scene are also finding more accessible, as Faye agrees.

> "There are certain things I know, and take for granted, that not all makers know"

"I sometimes feel a little behind because I haven't been making in an intense way my whole life...but I have been making. I have been doing it in some way, shape, or form in my spare time. The problem is that I never focused on a single thing; I never put it out there. It was always just a bit of fun when I could, and now I feel like I'm doing a bit of catch-up, catching up on lost time. But it is turning out to be an incredible thing. I'm very grateful to my friend."

So, Faye built a brand and began to turn her passion into her profession, producing YouTube videos, content for Instagram, and dedicating time to socialising with the online maker community, and her Patreons, as Geeky Faye Art.

"I came up with the idea of creating a brand that I could have fun with and enjoy being, that was me, but also still an online persona," she explains, "And at this point, I had already discovered 3D printing, and I'm very grateful for that because it's the coolest thing in the world."

3D printing is what the community will know Faye best for, as she continues to produce more online content for her various social accounts. From printing and building beloved cartoon characters such as SpongeBob's pet snail, Gary, to designing and printing her own ideas, Faye's YouTube channel has the feel of an artist with a passion for 3D printing and design.

DESIGNS DRAWN FROM EXPERIENCE

"There are certain things I know, and take for granted, that not all makers know. Because a lot of makers come from different backgrounds, from engineering and science, for example. I come from an art background. I'm a colour nerd; I get excited about colour. I realise I know things about colour and composition and perspective – a lot of the fundamentals of art that some makers don't know but I do, and I take it for granted. It was part of my education; it's part of my background."

Another use of her artistic background came in the form of a commission to design 3D-printable board game pieces for Bristol-based Riverward Games. "The pieces are 1920s-styled, so they were fashioned after *The Great Gatsby*," she explains. "They're resin, and quite tiny, but I put so much detail into them that I had to do some big prints too." The models were designed using ZBrush, a piece of software that has changed considerably since Faye last used it. "I hadn't used it in about three upgrades, so I was a bit like, 'what, what's all this new stuff?' But, if I'm not mistaken, Blender can do something →

Right
You can purchase design files to print your own picture frame at geekyfaye.art/digital-shop/

very much like this." Which is good for the rest of us as, unlike ZBrush, Blender is free to download.

But that isn't the only original piece Faye has put out into the world for public consumption. In December last year, she released her most challenging product to date, the Geeky Faye Art Ultimate Picture Frame, a 3D-printable framing system that would allow users to print picture frames in any size they needed (**hsmag.cc/UltimateFrame**).

"I made it to save money," she begins. "I had a bunch of things to frame. Some in inches, some were metric, some were A-sized, some were B-sized. That was too many expensive custom frames to buy. And that was the idea behind the system. I wanted to be cheap. I wanted to make one design, and yeah, I wanted to be lazy. Laziness is the mother of invention."

So, Faye began the task of creating the system that would print frames in centimetres, inches, A-size, and B-size, allowing for as simple a user experience as possible. And it works. Between the clear labelling of her files, to the custom-built sizing system on her website, the frames are unlike anything else currently available to makers – (**geekyfaye.art/digital-shop**).

"When people have seen it, their minds have been blown," she beams. "And the number of people who I respect personally who have seen it and have been shocked, that really does say something to me. Because, when I was working it, making it and printing it, and putting it together, I was thinking, 'Wow, this is working, this is really cool.' I felt like I was onto something." Faye's excitement for the product is obvious. She's unable to hold back the smiles as she explains further.

"I have never designed a product for mass production before. It's not something I have experience in. But I have experience in all these other things, so I thought, 'Why not? Just give it a go'. That should be my motto. I'd figured out all the bits that I did know, and I could figure the rest out as I go."

The frames have been well received, and have already started to make →

Left
Some of Faye's designs are on Thingiverse: hsmag.cc/3mYl3j

Left
With resin printers, you can get a lot of detail in a small space

appearances in videos by online creators Faye respects.

IN THE WILD

"It's been a slow burn, but I love seeing people using it, sharing their prints. Seeing people like Becky Stern using it and making it her own, I was excited (**hsmag.cc/BeckyStern**). That's what I want. I want people to take it, customise it, make it into their own, use it in an unexpected way. I wanted to make a design that can be used for almost anything, that you can make it any size, any use case. Try me, see what you can find that doesn't work."

And if it doesn't work?

"I've been working on some add-ons. I've had someone get in touch who had a *Star Wars* banner they wanted to frame that was 94 inches by 12 inches, and he commissioned me to design supports to add stability." This freedom to add new pieces to the pre-existing model, and to amend it as users see fit, it's what makes the system all the more interesting. And the ever-dropping prices of affordable 3D printers make the Ultimate Picture Frame a viable replacement for off-the-shelf frames, especially if your prints are odd sizes. Another tick in the pro box for at-home 3D printing since, as Faye states in the video for her system, "3D printing is awesome because it turns a designer like myself into a full production facility, able to pluck ideas out of my brain and make them into real physical reproducible things in a matter of hours or days." Well, maybe not days in this instance, but the time spent testing and retesting her designs only added to the success of the final product.

"It was a lot of maths and a lot of testing, that's what took so much time as I only had one printer," she laughs, but this hindrance was a blessing, as most of the people using the frame system would be in a similar situation. "So I was like, 'OK, how can people print this regardless of the tolerances of their printer?' " So, she made versions of the frame for different tolerances, with printable test-pieces to save makers time (and filament) choosing the best option. "I learned a lot about user experience. Yes, it was 3D design and 3D printing, but it was also user experience."

DESIGNING FOR OTHER PEOPLE

"Accessibility is often something I harp on about and is something that is often missing in the 3D printing community. I can't tell you the number of files I've downloaded where the design is cool but the designer has put no effort into the user experience. I downloaded a cookbook stand. It was a beautiful design, it was multiple parts, and it all came in as a single STL file. I spent two hours breaking it down into pieces. There were no instructions, no BOM (bill of materials), nothing. To me, that's important."

This ideology isn't just for online files. Faye spent hours working, with help, to create an easy-to-use interface for picking the right pieces to print. Simply insert your image size into the generator and it'll tell you which pieces to print and how many of them, providing the smoothest of customer experiences.

But it's not all 3D printing and picture frames. Faye also co-hosts a segment on fellow 3D printing maker Billie Ruben's YouTube channel. When time zones aren't in their way, the two get together online to interview other members of the maker industry, such as the wrangler of robot sidekicks, Jorvon Moss, the ever-inspiring Hannah Makes, and Lenore Edman, co-founder of Evil Mad Scientist Laboratories (**hsmag.cc/BillieRuben**).

"Billie is the nicest person. She's a great friend. And we were trying to think of how we can do cool things together. How can we work together when we're on the other side of the planet to each other? And when the 'Meet a Maker' idea was suggested to her by one of her Patreons, her first thought was, 'This would be so fun to do with Faye.' So she brought the idea to me, and we were both super-excited to do it, and then the next day, we realised how hard it would be. Sometimes it's impossible to coordinate so many time zones, so I'm not always there. I make an effort to be there as much as I can, and Billie always makes the effort to include me."

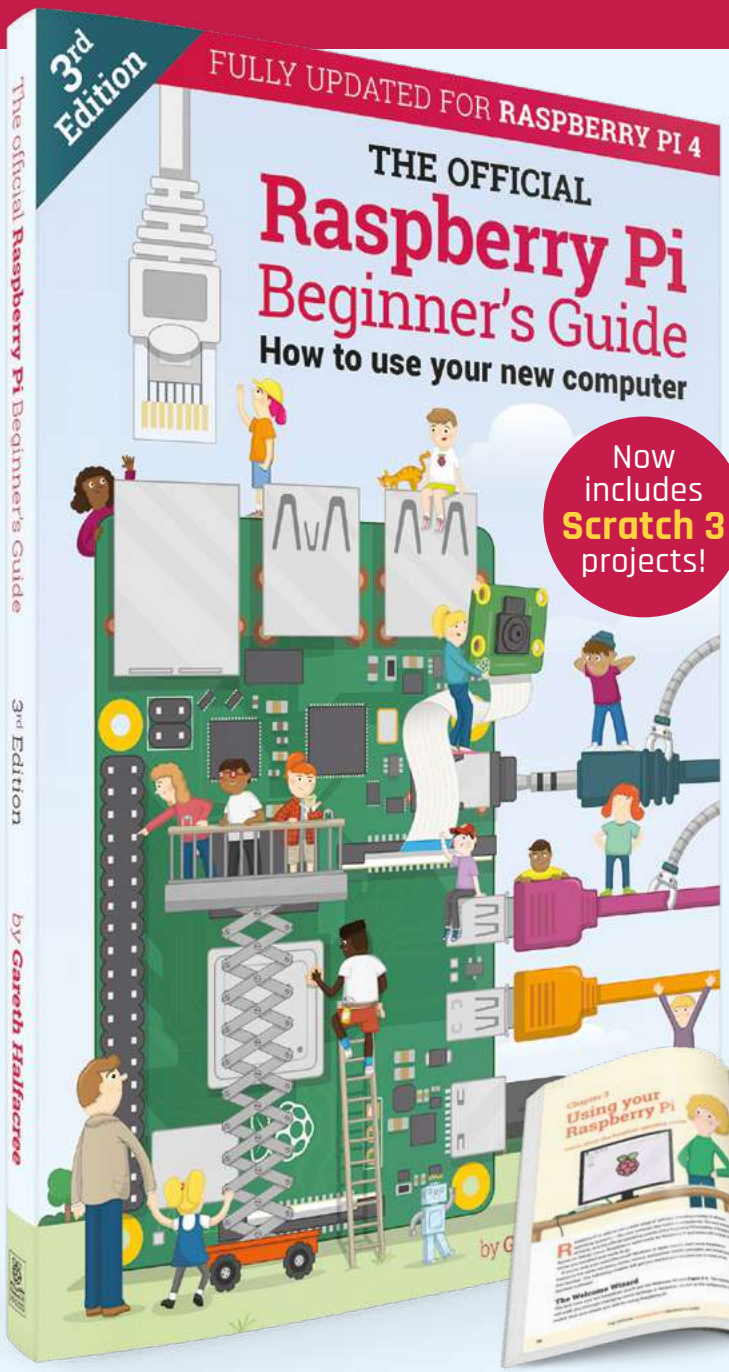
As more and more people start to involve video calls in their content collaborations, the two have definitely had their fair share of teething issues.

"We always joke when we're getting set up that we're cursed because something will always go wrong. But, as we're working through the kinks and finding our sea legs with it, it's turning out to be a really fun little show. As it's on her channel, she does most of the work, so I get to show up and talk to interesting people, mostly with my jaw on the floor because everyone is so incredible and so inspiring. I'm surprised no one has made a set of GIFs of my facial reactions. Billie made it happen, and I'm just so incredibly happy and fortunate to call her a friend. People call her the nicest maker on the internet, and it's true."

"I'm aware I'm a little older than some of these kids getting into making," Faye summarises. "And I'm very happy to be able to put out there that it's never too late to reinvent yourself." And, for our readers, we hope this inspires more of you to share your makes and explore your interests. ◻

Below
You can find Faye on Twitter (@GeekyFayeArt), and Instagram (instagram.com/geekyfayeart), and follow Geeky Faye Art on YouTube (hsmag.cc/GeekyFaye) for more content. And as for those customisable frames? Check out her website (geekyfaye.art) and try them for yourself

THE OFFICIAL Raspberry Pi Beginner's Guide



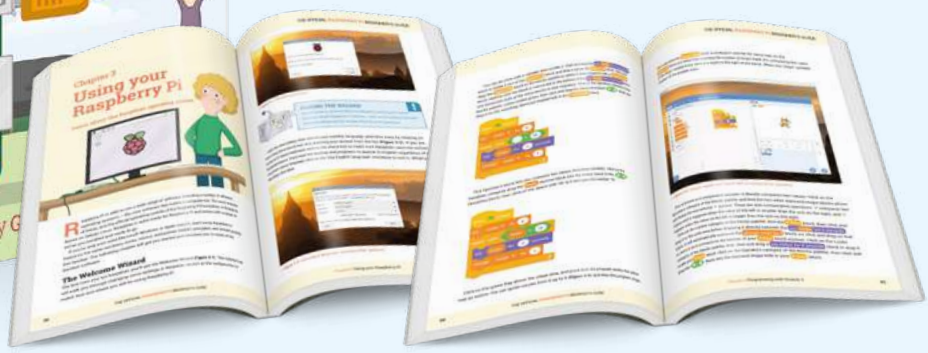
The only guide you need to get started with Raspberry Pi

Inside:

- Learn how to set up your Raspberry Pi, install an operating system, and start using it
- Follow step-by-step guides to code your own animations and games, using both the Scratch 3 and Python languages
- Create amazing projects by connecting electronic components to Raspberry Pi's GPIO pins

Plus much, much more!

£10 with FREE worldwide delivery



Buy online: magpi.cc/BGbook

FEATURE

IMPROVISER'S TOOLBOX

ZIPS

Fasten your builds

Mayank Sharma

@geekybodhi

Mayank is a Padawan maker with an irrational fear of drills. He likes to replicate electronic builds, and gets a kick out of hacking everyday objects creatively.

The zip (or zipper to our transatlantic friends) is one of the most common and convenient means to fasten all kinds of items, from garments to luggage. In fact, Global Industry Analysts, Inc. estimates the global market for zips at £11 billion in the year 2020. It predicts this figure to reach £13.5 billion by 2027.

A zip is essentially a type of an interlocking geared mechanism. The zip's teeth act as the cogs of the gear. The slider aligns the teeth to close the zip, allowing each tooth to mesh with the next tooth as they are brought together. Inversely, when you open a zip, a diamond-shaped wedge in the interior of the slider is forced between the teeth, which causes them to split apart and open.

Perhaps one of the most interesting aspects about the zip is the name itself. The word 'zip' is an onomatopoeic, which are words that are created because of the sound that's made when it's used, in this case, a high-pitched zip.

> "These days, a whopping majority of zips are manufactured by the Japanese YKK manufacturing company"

Before zips came along, buttons were the main fasteners for clothing and shoes. There have been several attempts to create a convenient sliding fastener. Elias Howe is often credited for patenting a design for one in 1851, which was never produced since he got busy with his other invention, the sewing machine.

Several decades later, Whitcomb Judson came up with something he called a clasp-locker, which garnered favourable reviews when it was showcased at the World's Columbian Exposition fair held in 1893 in Chicago.

But the zip, as we know it, didn't hit the shelves until 1914, when an engineer named Gideon Sundback perfected the previous designs for his slide fastener. The military was the first to adapt the zip at a large scale, for windproof flying suits, before the BFGoodrich tire company put one on a new type of rubber boot it introduced in the 1920s.

It took another couple of decades for zips to make their way to clothing. The credit goes to a number of French fashion designers who started using zips in their designs in 1937. In 1954, Levi's introduced a special zipped version of its overalls, before switching to it across their entire line of jeans in the 1970s.

These days, a whopping majority of zips are manufactured by the Japanese YKK manufacturing company. In fact, chances are you'll see YKK embossed on the pull of the zip in the clothes you've got on.

Here are some improvised uses of zips that we find particularly useful.

Zip-up laces

After labouring in her maker's den all day, Brooke finds it exhausting to unlace her shoes. So, she decided to replace the laces with zips instead. Her Instructable shows the steps she's used for modding high-tops, but she's also used the process on her husband's pair of boots. She begins by marking the shoe's eyelets on one side of the zip tape, before cutting them out and fixing the eyelets using specialised eyelet pliers. You then repeat the process on the other side of the zip tape by marking it through the eyelets on the first side. Recreate the eyelet-laced

> "She's also used the process on her husband's pair of boots"

zip tape for the other shoe. To attach the eyelet-laced zips to the shoes, Brooke used a pair of rivets. She folds over the bottom of the zips and then punches a pair of holes for the rivets that she then snaps on using a riveter. However, if you don't have a riveter or don't want to use rivets, Brooke suggests you also stitch the zip tapes to the shoes. Finally, use the eyelets to lace-up the shoes and tie any excess laces in the back. If that feels uncomfortable, you can instead tie them in a knot on each side of the zip to keep them snug. →

Project Maker
Brooke Brei

Project Link
hsmag.cc/ZipUP

Left ◆
Brooke's zip-up laces will be very convenient for anyone who has trouble handling laces, either due to injury or age

PENCIL POUCH

Project Maker
Joanne Loh

Project Link
hsmag.cc/PencilPouch

If you're handy with a needle, you'll love Joanne Loh's nifty little pencil pouch made from a length of ribbon and nylon zip tape. Both should be of the same length, and remember to add a zip pull to the zip before you begin. Joanne has illustrated each step of the process which, unless you're an excellent needle craftsperson, will take a couple of attempts. Begin by sewing a few stitches to make a bottom-stop about one inch from the end of the zip. Then tie a couple of knots to prevent the zip pull from accidentally coming off from the zip. Now back-stitch the ribbon to the zip tape. Joanne suggests you stitch a few centimetres away from the teeth to ensure the zip pull can glide smoothly. When you reach the bottom-stop, bend the zip tape downwards, cross the zip ends, and continue sewing. Follow the instructions carefully to ensure you get a nice smooth curve. You'll have to slip-stitch the ribbon to the tape after folding the ribbon's end to create the curve. When you're done, you'll end up with a pencil case that closes by zipping in an upward cycle. Use a different sized ribbon and zip tape to increase or decrease the size of the pouch.

Right
The pencil pouch builds on another of Joanne's zip crafts – a tetrahedron coin purse

JAZZED-UP CHARMS

Prolific maker and self-confessed DIYer, Shazni has used zips from old garments to jazz up some charms. She begins by cutting the metal zips away from the fabric and then burns the frayed edges. She's then used them to create brooches, pendants, and earrings, with some pieces of felt and glue. Her illustrated Instructable is fairly easy to follow. For the swirl pendant, she's sewed the zip to a piece of felt in a swirly pattern such that the thread doesn't show. Create swirls of different sizes and combine them into a pendant or a brooch. For her second design, she's arranged and glued different coloured zips into the shape of a dragonfly, and then used pieces of felt to fill in the body. For the third, she's used the zip to outline the shape of a leaf cut out on the felt. You can now glue in a brooch pin or a loop to use them as you see fit.

Project Maker
SHAZNI

Project Link
hsmag.cc/ZipCharms

Left ◈
Shazni suggests using a thick and stiff felt for the charms, but if you don't have one, she's shared a tip to make one without much effort

GIFT BOX

Want to use a zip to fasten something different? After sipping the water from a coconut on a particularly hot day, Bhawya decided to use the empty shells to create a gift box to store delicate knick-knacks. No points for guessing her fastener of choice for her coconut box! The trickiest bit of the build is to break the coconut into even pieces, which Bhawya suggests is best done with a hacksaw. Then scrape the meat off the shells, before sanding them on the outside. Now use a length of zip tape equal to the circumference of the shell, and stitch the ends. Finally, glue the zip tape to the inside of one shell, ensuring you leave some space for the pull. When that's done, unzip the zip and glue the other end to the inside of the other shell. Wait for it to dry, and you're done. □

Project Maker
BHAWYA

Project Link
hsmag.cc/CoconutZip

Left ◈
You can even paint the coconut shells to give it more funk, and perhaps even string a chain and turn it into a punk sling bag

SUBSCRIBE TODAY

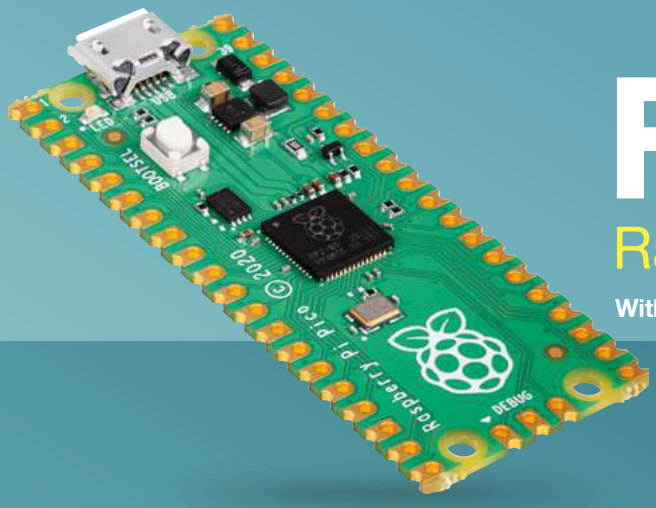
Get 12 issues of HackSpace magazine delivered to your door for just

£55 (UK)

£90 (USA)

£80 (EU)

£90 (Rest of World)



FREE!

Raspberry Pi Pico

With your first 12-month print subscription

This is a limited offer. Not included with renewals.
Offer subject to change or withdrawal at any time.

Subscribe online: hsmag.cc/subscribe

Subscription starts with next issue

HackSpace
TECHNOLOGY IN YOUR HANDS

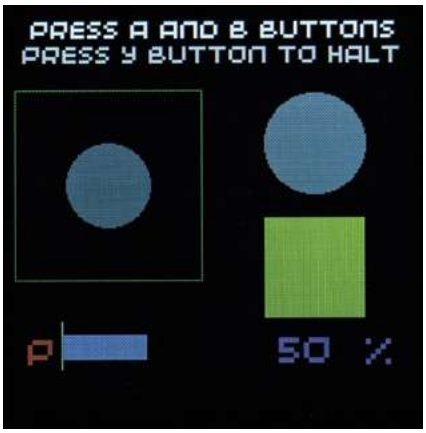
FORGE

HACK | MAKE | BUILD | CREATE

Improve your skills, learn something new, or just have fun tinkering – we hope you enjoy these hand-picked projects

Graphic routines for Pico screens

Code extra graphics grunt for your Pico Explorer



Above
The four buttons give you a way of getting data back from the user as well as displaying information

Tony Goodhew

Tony is a retired teacher of Computing and he has been helping people to code since 1968. He started with FORTRAN IV on an IBM 1130 with Hollerith cards for input.

Above
You don't have to get creative with your text placement, but you can

Pimoroni has brought out two add-ons with screens: Pico Display and Pico Explorer. A very basic set of methods is provided in the Pimoroni UF2 file. In this article, we aim to explain how the screens are controlled with these low-level instructions, and provide a library of extra routines and example code to help you produce stunning displays.

You will need to install the Pimoroni MicroPython UF2 file on your Pico and Thonny on your computer.

All graphical programs need the following 'boilerplate' code at the beginning to initialise the display and create the essential buffer. (We're using a Pico Explorer – just change the first line for a Pico Display board.)

```
import picoexplorer as display
# import picodisplay as display
#Screen essentials
width = display.get_width()
height = display.get_height()
display_buffer = bytearray(width * height * 2)
display.init(display_buffer)
```

This creates a buffer with a 16-bit colour element for each pixel of the 240×240 pixel screen. The code invisibly stores colour values in the buffer which are then revealed with a `display.update()` instruction.

The top-left corner of the screen is the origin (0,0) and the bottom-right pixel is (239,239).

SUPPLIED METHODS

```
display.set_pen(r, g, b)
```

Sets the current colour (red, green, blue) with values in the range 0 to 255.

```
grey = display.create_pen(100,100,100)
```

Allows naming of a colour for later use.

```
display.clear()
```

Fills all elements in the buffer with the current colour.

```
display.update()
```

Makes the current values stored in the buffer visible. (Shows what has been written.)

```
display.pixel(x, y)
```

Draws a single pixel with the current colour at point(x, y).

```
display.rectangle(x, y ,w ,h)
```

Draws a filled rectangle from point(x, y), w pixels wide and h pixels high.

```
display.circle(x, y, r)
```

Draws a filled circle with centre (**x**, **y**) and radius **r**.

```
display.character(78, 112, 5 ,2)
```

Draws character number 78 (ASCII = 'N') at point (112,5) in size 2. Size 1 is very small, while 6 is rather blocky.

```
display.text("Pixels", 63, 25, 200, 4)
```

Draws the text on the screen from (63,25) in size 4 with text wrapping to next line at a 'space' if the text is longer than 200 pixels. (Complicated but very useful.)

```
display.pixel_span(30,190,180)
```

Draws a horizontal line 180 pixels long from point (30,190).

```
display.set_clip(20, 135, 200, 100)
```

After this instruction, which sets a rectangular area from (20,135), 200 pixels wide and 100 pixels high, only pixels drawn within the set area are put into the buffer. Drawing outside the area is ignored. So only those parts of a large circle intersecting with the clip are effective. We used this method to create the red segment.

```
display.remove_clip()
```

This removes the clip.

```
display.update()
```

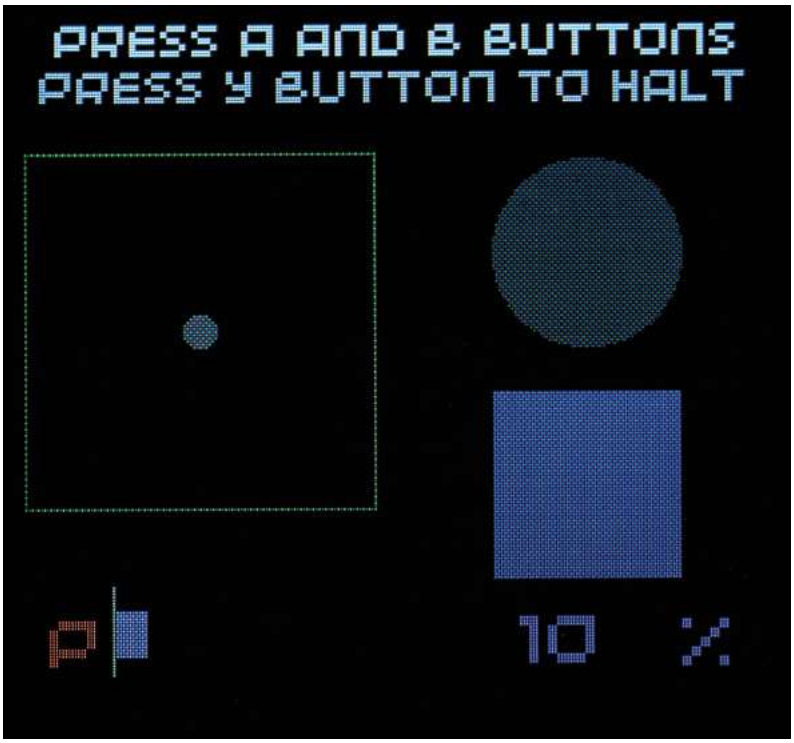
This makes the current state of the buffer visible on the screen. Often forgotten.

```
if display.is_pressed(3): # Y button is pressed ?
```

Read a button, numbered 0 to 3.

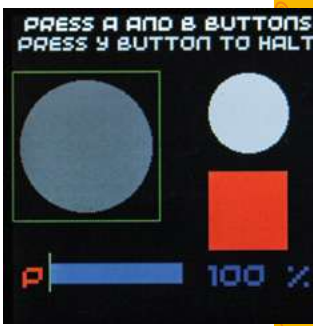
This code demonstrates the built-in methods and can be downloaded here: hsmag.cc/issue41

```
# Pico Explorer - Basics
# Tony Goodhew - 20th Feb 2021
import picoexplorer as display
import utime, random
#Screen essentials
width = display.get_width()
height = display.get_height()
display_buffer = bytearray(width * height * 2)
display.init(display_buffer)

def blk():
    display.set_pen(0,0,0)
    display.clear()
    display.update()

def show(tt):
    display.update()
    utime.sleep(tt)

def title(msg,r,g,b):
    blk()
    display.set_pen(r,g,b)
    display.text(msg, 20, 70, 200, 4)
    show(2)
    blk()

# Named pen colour
grey = display.create_pen(100,100,100)
# ==== Main ======
blk()
title("Pico Explorer Graphics",200,200,0)
display.set_pen(255,0,0)
display.clear()
display.set_pen(0,0,0)
```



Above
While the screens are quite small in size, they have plenty of pixels for display

Below
You can get more creative with the colours if you wish



Above
Straight lines can give the appearance of curves

```
display.rectangle(2,2,235,235)
show(1)
# Blue rectangles
display.set_pen(0,0,255)
display.rectangle(3,107,20,20)
display.rectangle(216,107,20,20)
display.rectangle(107,3,20,20)
display.rectangle(107,216,20,20)
display.set_pen(200,200,200)
#Compass  points
display.character(78,112,5,2)   # N
display.character(83,113,218,2) # S
display.character(87,7,110,2)   # W
display.character(69,222,110,2) # E
show(1)
# Pixels
display.set_pen(255,255,0)
display.text("Pixels", 63, 25, 200, 4)
display.set_pen(0,200,0)
display.rectangle(58,58,124,124)
display.set_pen(30,30,30)
display.rectangle(60,60,120,120)
display.update()
display.set_pen(0,255,0)
for i in range(500):
    xp = random.randint(0,119) + 60
    yp = random.randint(0,119) + 60
    display.pixel(xp,yp)
    display.update()
```

```
show(1)
# Horizontal line
display.set_pen(0,180,0)
display.pixel_span(30,190,180)
show(1)
# Circle
display.circle(119,119,50)
show(1.5)
display.set_clip(20,135, 200, 100)
display.set_pen(200,0,0)
display.circle(119,119,50)
display.remove_clip()

display.set_pen(0,0,0)
display.text("Circle", 76, 110, 194, 3)
display.text("Clipped", 85, 138, 194, 2)
display.set_pen(grey) # Previously saved colour
# Button Y
display.text("Press button y", 47, 195, 208, 2)
show(0)
running = True
while running:
    if display.is_pressed(3): # Y button is pressed
?
        running = False
blk()

# Tidy up
title("Done",200,0,0)
show(2)
blk()
```

We've included three short procedures to help reduce code repetition:

```
def blk()
```

This clears the screen to black – the normal background colour.

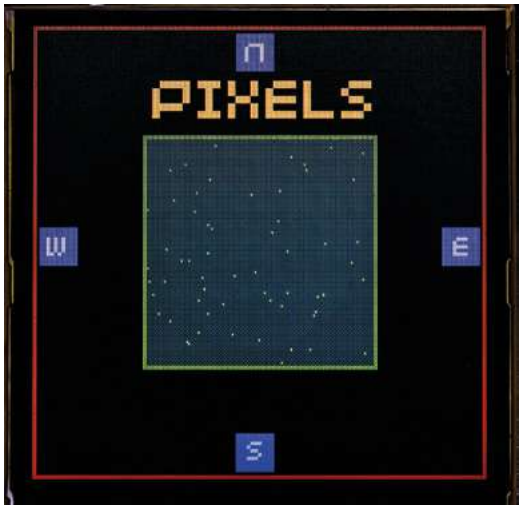
```
def show(tt)
```

This updates the screen, making the buffer visible and then waits tt seconds.

```
def title(msg,r,g,b)
```

This is used to display the `msg` string in size 4 text in the specified colour for two seconds, and then clears the display.

As you can see from the demonstration, we can accomplish a great deal using just these built-in methods. However, it would be useful to be able to draw vertical lines, lines from point A to point B, hollow circles, and rectangles. If these are written as procedures, we can easily copy and paste them into new projects to save time and effort.



In our second demonstration, we've included these 'helper' procedures. They use the parameters (t, l, r, b) to represent the (top, left) and the (right, bottom) corners of rectangles or lines.

```
def horiz(l,t,r):    # left, top, right
```

Draws a horizontal line.

```
def vert(l,t,b):   # left, top, bottom
```

Draws a vertical line.

```
def box(l,t,r,b):  # left, top, right, bottom
```

Draws an outline rectangular box.

```
def line(x,y,xx,yy):
```

Draws a line from (x,y) to (xx,yy).

```
def ring(cx,cy,rr,rim): # Centre, radius,
thickness
```

Draws a circle, centred on (cx,cy), of outer radius rr and pixel thickness of **rim**. This is easy and fast but has the disadvantage that it wipes out anything inside **ring**.

```
def ring2(cx,cy,r):   # Centre (x,y), radius
```

Draw a circle centred on (cx,cy), of radius rr with a single-pixel width. Can be used to flash a ring around something already drawn on the screen. You need to **import math** as it uses trigonometry.

```
def align(n, max_chars):
```

This returns a string version of int(n), right aligned in a string of **max_chars** length. Unfortunately, the font supplied by Pimoroni in its UF2 is not monospaced.

The second demonstration is too long to print, but can be downloaded here: hsmag.cc/issue41

It illustrates the character set, drawing of lines, circles and boxes; plotting graphs, writing text at an angle or following a curved path, scrolling text along a sine curve, controlling an interactive bar graph with the buttons, updating a numeric value, changing the size and brightness of disks, and the colour of a rectangle.

The program is fully commented, so it should be quite easy to follow.

The most common coding mistake is to forget the **display.update()** instruction after drawing something. The second is putting it in the wrong place.

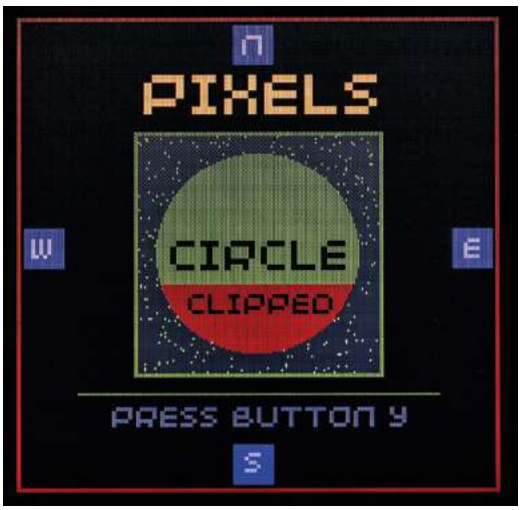
When overwriting text on the screen to update a changing value, you should first overwrite the value with a small rectangle in the background colour. Notice that the percentage value is right-aligned to lock the 'units' position.

It's probably not a good idea to leave your display brightly lit for hours at a time. Several people have reported the appearance of 'burn' on a dark background, or 'ghost' marks after very bright items against a dark background have been displayed for some time. We've seen them on our display, but no long-term harm is evident. Blanking the screen in the 'tidy-up' sequence at the end of your program may help.

> "It's probably not a good idea to leave your display **brightly lit for hours at a time**"

We hope you have found this tutorial useful and that it encourages you to start sending your output to a display. This is so much more rewarding than just printing to the REPL.

If you have a Pimoroni Pico Display, (240×135 pixels), all of these routines will work on your board. □



Left
What will you create with your Pico display? Let us know! hsmag.cc/hello

Left
You don't need much to create interesting graphics

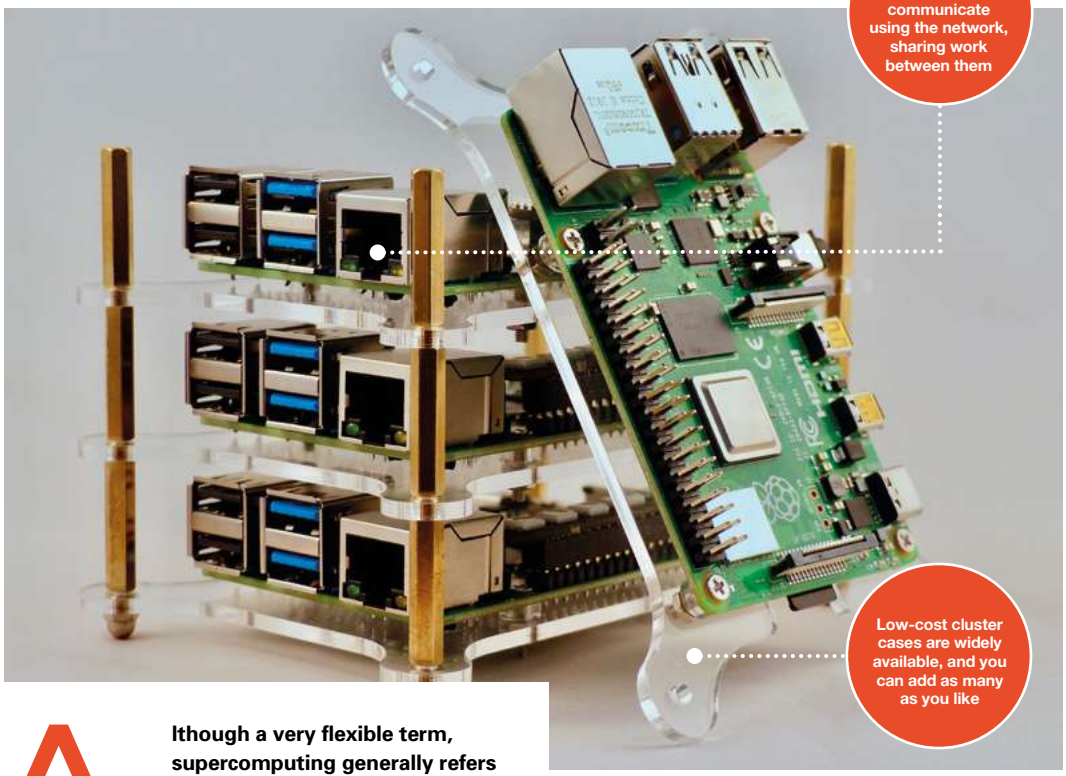
Supercomputing with Raspberry Pi

What's better than one Raspberry Pi Computer? Lots of Raspberry Pi computers! PJ Evans checks out some amazing projects and explains how to build your own clustered supercomputer

PJ Evans

@MrPJEvans

PJ Evans is a developer and wrangler of the Milton Keynes Raspberry Jam. He runs a LoRa gateway, which is probably the nearest he'll get to his own radio breakfast show.



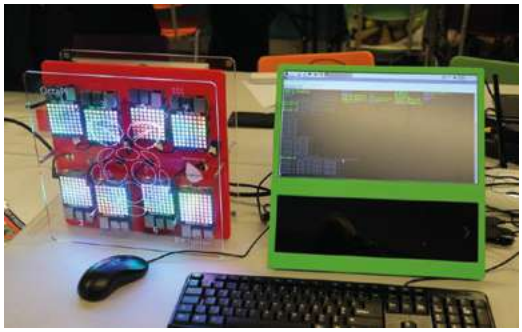
Although a very flexible term, supercomputing generally refers to the idea of running multiple computers as one, dividing up the work between them so that they process in parallel. In theory, every time you double the amount of processing power available, you half the time needed to complete your task. This concept of 'clusters' of computers has been implemented heavily in large data processing operations, including intensive graphics work such as Pixar's famous 'render farm'. Normally the domain of large organisations, supercomputing is now inthe hands of the masses in the form of education projects and makes from the cluster-curious, but there have also been some impressive real-world applications. Here, we'll look at some amazing projects and get you started on your own supercomputing adventure.

QUICK TIP

If you're building your own supercomputer, always take power and cooling into consideration. Raspberry Pi 4 runs hotter and consumes more power than its older siblings.

OCTAPI

One of the first high-profile cluster projects surprisingly came from the boffins at GCHQ (Government Communications Headquarters) in the UK. Created as part of their educational outreach programme, the OctaPi used eight Raspberry Pi 3B computers to create a cluster. Kits were loaned out to schools with multiple coding projects to engage young minds. The first



demonstrated how supercomputing could speed up difficult equations by calculating pi. A more advanced, and very appropriate, task showed how these eight machines could work together to crack a wartime Enigma code in a fraction of the time it would have taken Bletchley Park.

hsmag.cc/OctaPi

TURING PI

As we've already said, most Raspberry Pi cluster projects are for education or fun, but there are those who take it seriously. The Raspberry Pi Compute Module form factor is perfect for building industrial-grade supercomputers, and that's exactly what Turing Pi has done. Their custom Turing Pi 1 PCB can accept up to seven Raspberry Pi 3+ Compute Modules and takes care of networking, power, and USB connectivity. Although claiming a wide range of uses, it appears to have found a niche in the Kubernetes world, being a surprisingly powerful device for its price. Future plans have been announced for the Turing Pi 2, based on the more powerful Raspberry Pi 4.

turingpi.com

WATER-COOLED CLUSTER

Multiple machines are one thing, but there's also the individual speed of those machines. The faster they go, the faster the cluster operates exponentially. Overclocking is common in supercomputing, and that means heat. This project, which maker Michael Klements freely admits is one of those 'just because' undertakings, uses the kind of water cooling usually found on high-end gaming PCs and applies it to a Raspberry Pi cluster. This beautiful build, complete with laser-cut mounts and elegant wiring, has been extensively documented by Klements in his blog posts. We can't wait to see what he does with it!

hsmag.cc/WaterCool

ORACLE SUPERCOMPUTER

So how far can we take this? Who has built the largest Raspberry Pi cluster? A strong contender seems to be Oracle, who showed off their efforts at Oracle OpenWorld in 2019. No fewer than 1060 Raspberry Pi 3B+ computers were used in its construction (that's 4240 cores). Why 1060? That's as much as they could physically fit in the frame! The creation has no particular purpose bar a demonstration of what is possible in a small space, cramming in several network switches, arrays of USB power supplies, and a NAS (network-attached storage) for boot images.

hsmag.cc/Oracle1060

MAKE YOUR OWN

We're thinking you probably don't fancy trying to beat Oracle's record on your first attempt, and would like to start with something a bit simpler. Our sister magazine, The MagPi, has published a cluster project you can make at home with any number of Raspberry Pi devices (although just one might be a little pointless). In this case, four Raspberry Pi 4B computers were assigned the job of searching for prime numbers. Each is assigned a different starting number, and then each adds four before testing again. This is handled by an open-source cluster manager, MPI (Message Passing Interface). A solid introduction to what is possible.

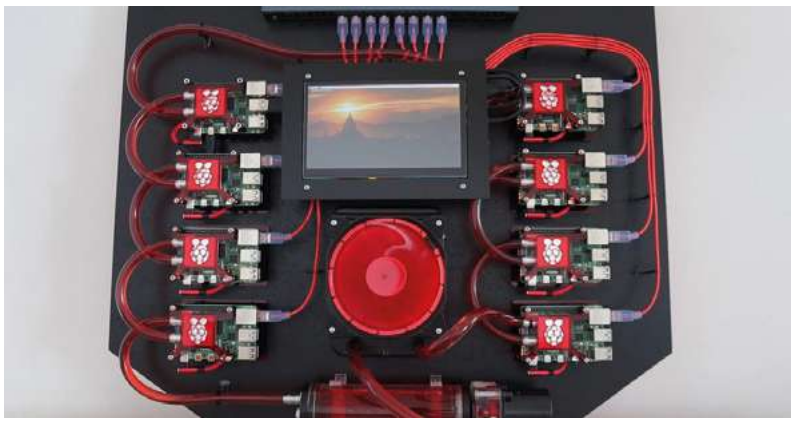
hsmag.cc/RPIcluster

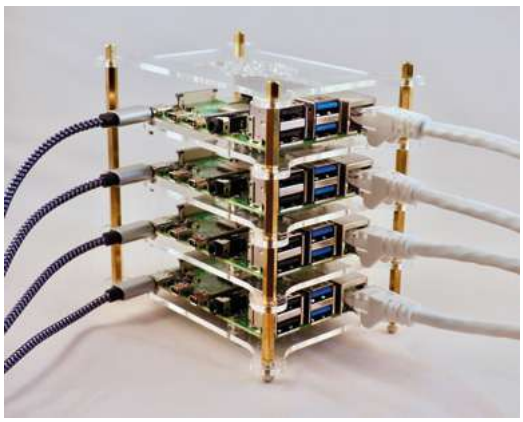
Above
OK, this is just getting ridiculous now

Left
Eight Raspberry Pis and a lot of LEDs

QUICK TIP

Programming languages such as R, which specialise in data analysis, have support for clusters baked-in. Make sure the software can use the hardware!

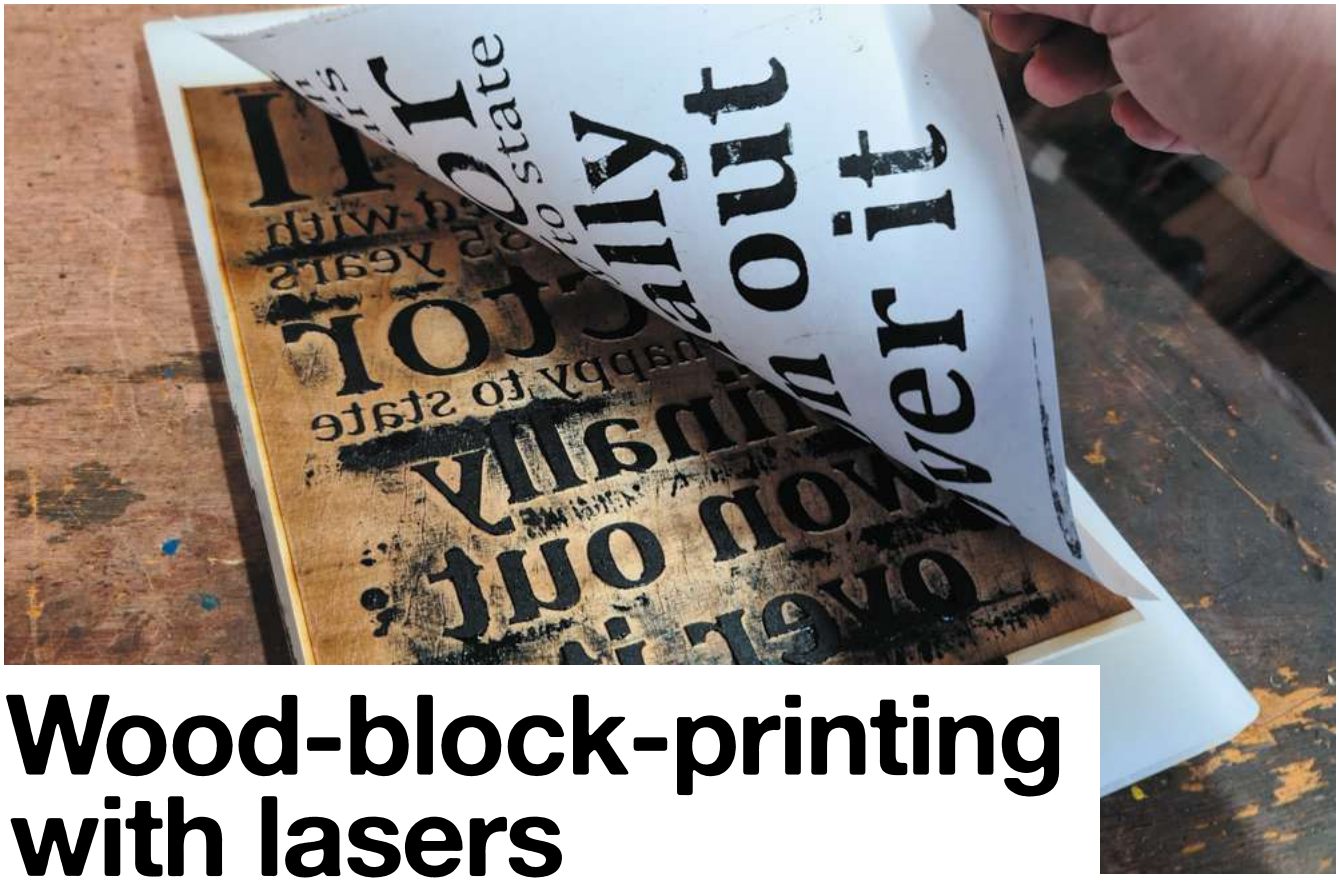




Below
Sometimes it's good to get water in your computer

Below
Cluster computing has never been more accessible

TUTORIAL



Wood-block-printing with lasers

Use a laser cutter to update 2000-year-old printing techniques

Dr Andrew Lewis

Dr Andrew Lewis is a specialist fabricator and maker, and is the owner of the Andrew Lewis Workshop.

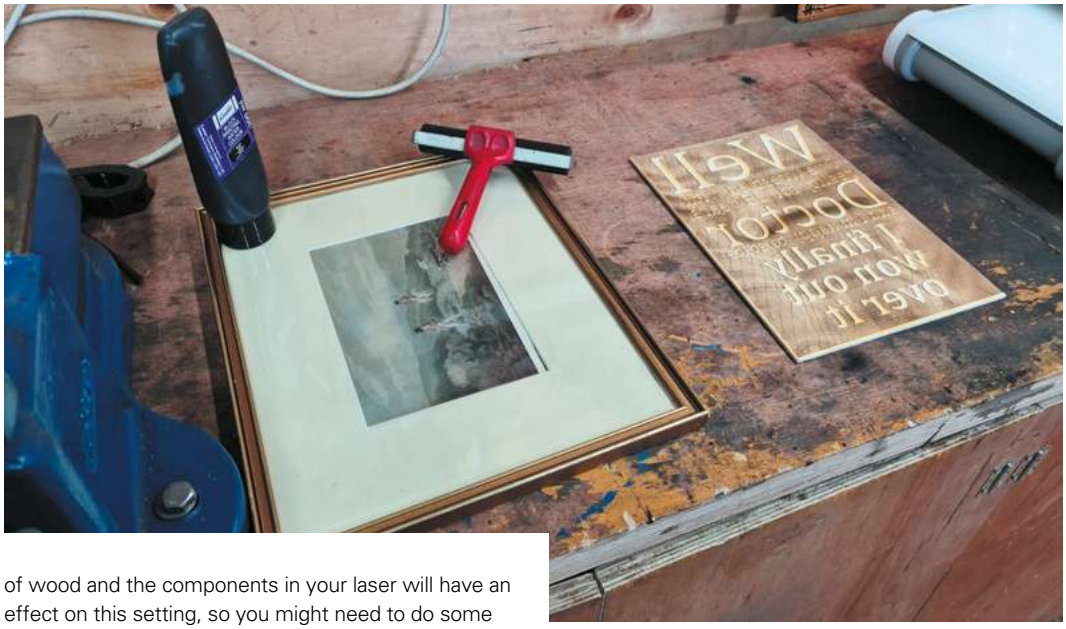
Above
Block-printing is a very satisfying process. Thanks to laser engraving, you can make your own printing blocks without having to carve the wood by hand

Printing from woodcuts (also known as block-printing) is one of the oldest printing techniques for paper and textiles. It's also one of the most versatile methods of printing, offering more opportunities for experimentation than other similar techniques like lino printing. Different combinations of wood grain and ink can create different effects on finished prints. If you're just starting out with printing, learning the skills needed to carve the block by hand can be a huge stumbling block, and having the right type of wood to carve cleanly can be expensive. If you're lucky enough to have access to a laser cutter, you can start experimenting much more quickly by using that to carve your wood-blocks.

There's plenty of room for experimentation with block-printing, but you can't just dive in without preparing your workspace. You'll need to be working in a clean, dry, well-lit space with flat surfaces and enough room to lay out your tools comfortably. You will also need somewhere to lay out wet prints while they dry, and plenty of cleaning supplies to deal with any spills as they happen.

The first step to creating your print is to figure out a suitable design. Wood has a grain, and the fineness of the grain will determine how much detail you'll be able to include in your print. Fine lines and dots are likely to break away from the block very quickly, particularly if you're using coarse-grained wood. Bold text and simple shapes are a nice starting point if you're nervous, but don't be afraid to push the limits of where you feel comfortable. You'll be raster etching the wood, so open up your favourite editor and get designing. If you want to find some inspiration, have a look at **oldbookillustrations.com**, and you can search for woodcuts from old books and get an idea of what's possible.

Transfer your design onto the wood-block using a laser cutter and K40 Whisperer. Remember that the printing process will reverse the printing plate, and that black areas of the image will be engraved into the wood. Chances are that you'll want to reverse the image and invert the colours so that your drawn lines will be proud of the wood, not engraved into it. Assuming that you're using a K40 laser, set the power to about 20%, and the speed to 200mm/s. Your choice



Left
Keeping your workspace free of ink splashes and dirt is important. You should leave plenty of space so that your glass sheet, printing plate, press, and paper aren't crowded on top of each other

Below
The more time you spend making sure your blank printing block is lined up in the laser, the better your engraving will be. If you have a slight misalignment at the edges of the block, you can sand or chisel off the high spots

of wood and the components in your laser will have an effect on this setting, so you might need to do some experimentation to get the best etch. It's worth taking your time setting up the wood in the printer, making sure that the focal distance and alignment are correct, and that you have plenty of airflow and ventilation in place to deal with the smoke created by woodcuts. Never leave the laser unattended when it's working,

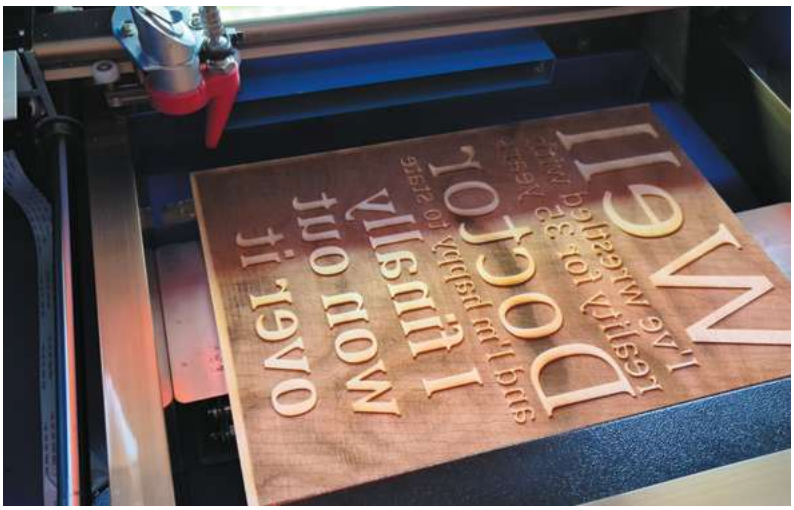
> **Transfer the roller over to the printing block,** and roll it over as evenly as you can

and check constantly to make sure everything is proceeding as planned.

The laser will probably boil out some sap from the wood and deposit some smoke stains, so your printed block will need a clean before it's used. A wipe with a solvent might be enough, or some gentle lapping with fine sandpaper might be better if your wood is a little bit uneven. If you're going to glue your block down to a larger piece of wood to keep it flat, now is the time to do it.

STAY FLAT

Thin pieces of wood will warp as they absorb ink, so thicker pieces are definitely preferable. Although you will be able to print with a warped plate, you'll find that the quality is poor because flexing wood will lead to uneven inking and will smudge the details as it passes through the roller or press. If you print on thin plywood, consider gluing it to a thicker piece of wood or plastic to hold it flat while printing.



Inking the block is an art in itself. Normally, you'll apply ink with a roller. It might seem like a trivial act, but loading the roller with an even amount of paint and controlling the pressure you apply from the roller to the block is something that takes a bit of practice. Start by applying some ink to a glass sheet. Block-printing ink is very thick, and it should just sit there on the glass without spreading out. Use a putty knife or spatula to get your ink spread roughly into a line, and then use the roller to start working it into an even coat on the glass. Use a piece of scrap paper or card to take any excess from the roller, and then roll it through the ink again while applying light pressure to take an even coat of ink onto the roller. Transfer the roller over to the printing block, and roll it over as evenly as you can. You want to avoid getting ink anywhere that it isn't supposed to be, particularly around the edges of the wood-block. →

YOU'LL NEED

- **A4 sheet of plywood, 6–18 mm in thickness**
- **Soft roller for inking**
- **Hard brayer, artist's press, or roller press**
- **Sheet of glass for rolling out ink**
- **Paper or fabric**
- **Block-printing ink or thick acrylic paint**

Right
Not all woods engrave the same on a laser engraver, and you might need to increase the power for harder woods. The Janka hardness of a wood is a way of measuring how resistant a wood is to damage. The hardness of pearwood, for example, is about 16400 newtons. This means that it is significantly harder than most plywood, and will need to be engraved at a higher power or at a slower rate to get a decent depth of cut. The higher the Janka hardness of a wood, the more power you will need to apply to engrave it. There's nothing wrong with using multiple passes with the laser to get a deeper cut, so take your time and don't be afraid to run the job again if you want a more prominent engraving

CARVE, INK, PRINT, REPEAT

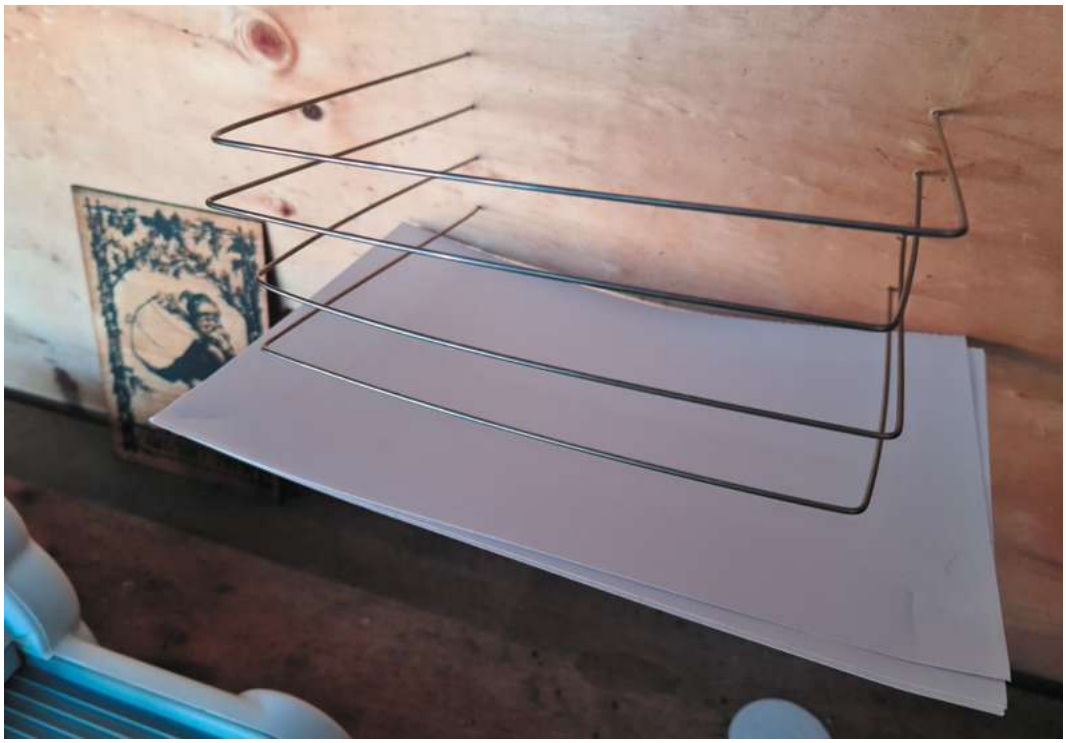
You should now have an inked block, and an excess of creative energy. Take a piece of thick paper and carefully lay it down on top of the printing block. It's important not to move the paper once it's touching the block, so a gentle touch and a little bit of practice might be needed to get your paper to sit straight. Apply another couple of sheets of thick paper on top of the first sheet. These sheets of paper serve a dual purpose. They protect the sheet you're printing with from any smudges or ink transfer from the back, but more importantly, they add some flexibility behind the printing sheet, allowing it to deform slightly into the printing plate and pick up more ink when pressure is applied. Using too many sheets will lead to smudging, but one or two sheets of paper will make the ink transfer more evenly onto your printed piece.

How you apply pressure to the back of the paper will depend on what tools you have available. If you have access to a book press or roller press, you can get an even result with minimal effort. You can also use a hard rubber brayer or even a rolling pin if that's all you have access to, but you'll need to practice applying even pressure and rolling across the back of the paper in a uniform way to get the best results. If you have a manual die cutter, like the Sizzix Big Shot,

PICK YOUR WOOD

The choice of wood you make for your printing block will affect the quality of the final prints you produce. It's fine to use plywood, but be aware that the printing block will be less crisp than it could be if you were using a more traditional block-printing wood like sycamore or pearwood.

Left
Block-printing ink is very sticky, and rolling it out makes it much easier to apply. Experiment with roller (also called brayer) hardness to get different effects. Softer rollers will cover uneven surfaces more effectively, at the expense of some accuracy. Small, hard rollers can be more effective for fine ink placement, allowing you to add multiple colours to a printing block in a single pass



Left
Some bent wire and a few small holes is all you need to stack your prints while they dry

Below
Block-printing is a great way of making custom cards or invitations for special occasions when an inkjet printer just doesn't seem personal enough

you can use layers of thin craft foam and plywood to shim the paper and printing plate so that it passes through the machine's fixed roller with sufficient pressure to transfer the ink when you turn the handle.

Carefully peel the printed sheet from the wood-block, and flip it over to admire your work. Place the paper somewhere flat to dry, or if you're stuck for space, hang it vertically on an indoor washing-line with pegs.

> **After you've made a few successful prints,** you'll probably start thinking about getting more adventurous

After you've made a few successful prints, you'll probably start thinking about getting more adventurous and using different colour combinations or techniques. You can apply multiple different colours to a single block using small rollers, or print multiple times onto the same sheet with different blocks. You can turn everything upside down, and use printing blocks as stamps to create patterns on textiles or other materials. If you're going to think about stamping, you can experiment with etching thin EVA foam glued to a wooden base to make a softer stamp. It isn't really wood-cutting at that point, but it is good fun!

PICK YOUR INK

Block-printing ink is very viscous, and there's good reason for that. However, it doesn't mean you can't use acrylics or oil paints to ink your blocks. You'll probably need to add a thickening agent to acrylic paints, but good-quality oil paints should be OK for printing. Be aware that oil paints take a long time to dry, and might not work well on all types of materials.

QUICK TIP

It's much easier to clean your rollers and other equipment while the ink is wet. Wash them clean as soon as you finish printing.

QUICK TIP

Block-prints are a bit like pancakes: the first couple always come out a bit weird. Take a few prints before you judge your work.

Make a digital do-not-disturb sign

Kids barging in on Zoom meetings? Parents being a pain when you're in Google Classroom? Let them know what you're up to with 'Busybot', the custom do-not-disturb sign

MAKER

PJ Evans

PJ is a writer, tinkerer, and software engineer. He spends all day on Zoom and no longer knows if his colleagues have legs or not.

@mrpjevans

There are often times when family and work life collide. The internet is littered with videos of online meetings being hijacked by partners, parents, and pets. We can't do anything about your cat, but maybe a digital sign can let people know what you're up to, and whether it's OK to wander in.

In this tutorial we're going to use a popular messaging protocol, MQTT (Message Queuing Telemetry Transport). With MQTT, we will set up a simple do-not-disturb sign that you can trigger with a single key press and customise to your heart's content.

You'll Need

- 2 × Raspberry Pi Zero W **magpi.cc/pizerow**
- Scroll pHAT HD **magpi.cc/scrollhathd**
- pHAT Diffuser **magpi.cc/phatdiffuser**
- Keybow Kit (3-key) **magpi.cc/keybow3**

01 Prepare Raspberry Pi

We're using two Raspberry Pi Zero W computers for this project, although it will work with any recent model. For both, we recommend installing Raspberry Pi OS Lite (**magpi.cc/software**) as we don't need a desktop interface. Make sure both devices are working and connected to the same network. We also recommend setting their host names by running `sudo raspi-config` at the command line, then going to System Options > Hostname. We chose 'busybot' for the sign and 'buttonbot' for the controller, and that's how we'll refer to them throughout this tutorial. Finally, make sure everything is up to date. Enter these commands:

```
sudo apt update
sudo apt full-upgrade
```

You should be able to ping one device from the other. On 'busybot' Raspberry Pi Zero W, enter:

```
ping buttonbot.local
```

You should get a response from 'buttonbot' Raspberry Pi Zero W.

02 Install the Scroll pHAT HD and diffuser

We're going to set up the display first. Pimoroni's Scroll pHAT HD is a display made up of 17×7 pixels (119 total) and comes with a Python library that does all the hard work of displaying and scrolling text for us. The pHAT form factor makes it just the right size for a display, but of course you can adapt this tutorial to any display you like. You'll need to solder the 40-pin header to the display and also add a reciprocal header to 'busybot' Raspberry Pi if it doesn't already have one. Before assembling them together, screw on the diffuser, which will make the display much easier to read. Now, with Raspberry Pi powered off, carefully connect the Scroll pHAT HD.

03 Set up the display software

It's time to make sure our display is working before we go any further. Fortunately, Pimoroni has created an install script for us. At the command line, run the following:

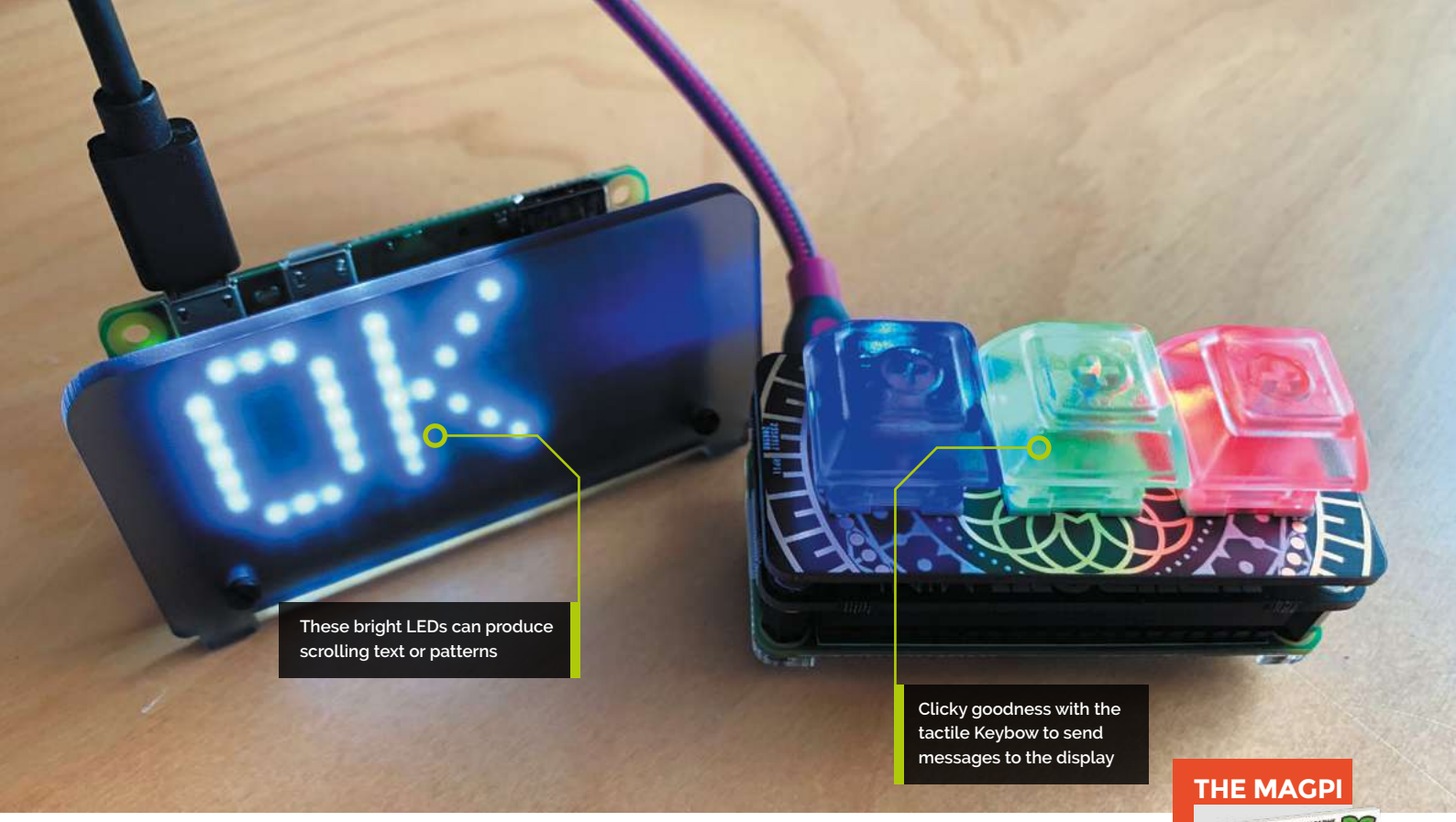
```
curl https://get.pimoroni.com/scrollphathd
| bash
```

Make sure you do the 'Full Install' when the option is presented. This will set up Raspberry Pi so it can communicate with the display and install all the Python libraries we need. Take a look at Pimoroni's GitHub (**magpi.cc/scrollphatgit**) for more information on installation.

Once the install has finished, we can test things out. Reboot, then try this on the command line:

```
cd ~/Pimoroni/scrollphathd/examples
python3 swirl.py
```

See a pretty pattern? Then you're ready to proceed. Have fun with the other examples in the directory; they're a great source of inspiration.



04 The interface

Our buttonbot and busybot are going to need to talk to each other over the network. One of the easiest, and most popular, ways to do this is the MQTT protocol. It uses a pub/sub model (publisher / subscriber) to process messages. An MQTT server (the 'broker') receives messages from 'publishers' that are then transmitted to 'subscribers'. These are organised by 'topic'. The biggest advantage is that publishers don't need to understand or even be aware of the subscribers, they just need to speak MQTT. Don't worry if this is confusing; working through the tutorial will make things clearer.

05 Mosquitto

For our system to work, you need an MQTT server (or 'broker') to handle the messages. This can be anywhere on your network, but for the purposes of the tutorial we'll install it on the same 'busybot' Raspberry Pi driving the display. MQTT software tends to be very 'lightweight', so a Raspberry Pi Zero W can easily handle being the server as well as a publisher. Mosquitto is probably the most popular set of MQTT tools. Installation is straightforward, too. Enter this at the command line:

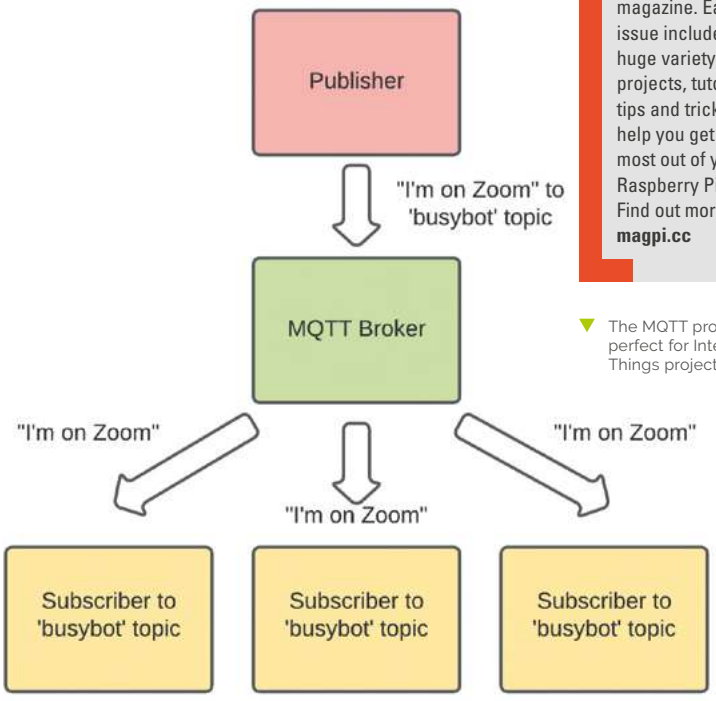
```
sudo apt install mosquitto mosquitto-clients
sudo pip3 install paho-mqtt
```

The broker will be automatically installed as a service and always be running in the background. We've also installed Paho, a popular Python library for implementing MQTT.

06 Connectors

The code for this project does two main jobs: listens to the MQTT server for new messages, and then takes those messages and scrolls them on the display. You can enter the code shown here or get the files from **magpi.cc/busybotgit**. We need the code to be running all the time. To do this, create a new file from the command line:

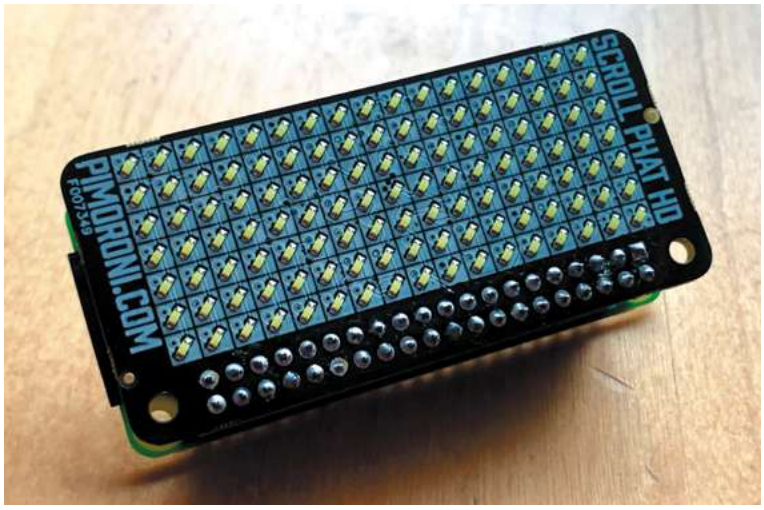
```
sudo nano /usr/lib/systemd/busybot.service
```



▼ The MQTT protocol is perfect for Internet of Things projects

THE MAGPI

This tutorial is from in The MagPi, the official Raspberry Pi magazine. Each issue includes a huge variety of projects, tutorials, tips and tricks to help you get the most out of your Raspberry Pi. Find out more at **magpi.cc**



▲ The Scroll pHAT HD without its diffuser. 119 very bright LEDs

Enter the code from **busybot.service**. (Change the paths if you've created the code elsewhere.) Save the file (**CTRL+X** and then **Y**), then enter these commands:

```
sudo systemctl enable /usr/lib/systemd/busybot.service
sudo systemctl start busybot
```

Top Tip

In the upside-down?

If your display is looking a little topsy-turvy, you can flip the display by uncommenting the line containing `scrollphathd.flip(1, 1)` in **busybot_magpi.py**.

07 Testing time

Let's confirm that the code is working. Using the Paho MQTT libraries, the code subscribes to the MQTT topic 'busybot' on the broker. Whenever anything publishes a line of text to that topic, busybot will be notified, the text delivered and then displayed. We can check everything is working using the Mosquitto command line publishing tool:

```
mosquitto_pub -h localhost -t busybot -m "Hello from MagPi"
```

busybot.service

Language: **Bash**

```
[Unit]
Description=busybot

[Service]
ExecStart=/usr/bin/python3 /home/pi/busybot/busybot_magpi.py
Restart=on-failure
User=pi
Group=pi

[Install]
WantedBy=multi-user.target
```

▲ Need more buttons? Try the nine-key Keybow

If you see the messages scrolling across, then everything is working. Send any message you wish, or a blank space to clear the display.

08 Assemble the Keybow

Now we have our display working, it's time to turn our attention to sending the messages. MQTT is widely supported and you can get clients for almost every platform and programming language in common use. That means we can send messages to the display from pretty much anywhere. In this tutorial, we're going to use Pimoroni's Keybow interface to provide a quick way of setting messages. On the second 'buttonbot' Raspberry Pi WH, assemble the Keybow as instructed at **magpi.cc/assemblekeybowmini**, but don't install Keybow OS – we'll stick with Raspberry Pi OS Lite.

> "Whenever anything publishes a line of text to that topic, busybot will be notified"

09 Keybow setup

We'll now add some code to send messages to the MQTT broker when buttons are pressed. First, from the command line, we need to install some dependencies on buttonbot:

```
sudo apt install python3-pip git
sudo pip3 install keybow paho-mqtt
```

Now get the code from GitHub:

```
cd
git clone https://github.com/mrpjevans/busybot.git
```

We can now test the Keybow with a simple example:

```
cd ~/busybot
python3 test_keybow.py
```

Press the keys. Do they all light up? Then all is well.

10 Keybow code

In the **busybot** directory, have a look at **buttonbot.py**. You'll also need to change the name of the MQTT broker and/or topic if you've used something different.

We need to make sure the code is always running, just like busybot. Again, we'll create a service to do this. From buttonbot's command line, go through the process in Step 6 to create a service file and enable it. Just make sure you change the `ExecStart` line to:

```
ExecStart=/usr/bin/python3 /home/pi/busybot/
buttonbot.py
```

Save the file and enable it as before.

11 Testing and tinkering

Everything should now be ready. With both Raspberry Pi Zero computers running, try pressing a key on the Keybow. A message will now scroll across the display on the other Raspberry Pi Zero. Make sure all three keys work. You're now ready to start customising the system to your own needs. If you edit **buttonbot.py**, you'll see some documented options for changing the messages and the colours of the keys. Free free to experiment, make changes, and make this code your own. If you need more than three messages, see if you can alter the code to support key press combinations, which would give you up to seven options.

12 Python vs C++

As the display doesn't 'know' about buttonbot's existence, it means that anything capable of speaking MQTT can send messages to busybot to set the scrolling text. It could be done from the command line or when a certain event happens. If you've been following the Home Assistant (HA) series in *The MagPi*, then you may be interested to know that HA speaks MQTT, so the display could be tied to a temperature/motion sensor. Have fun dreaming up new ideas.

busybot_magpi.py

DOWNLOAD THE FULL CODE:
magpi.cc/busybotgit

Language: **Python 3**

```
001. import time
002. import scrollphathd
003. import paho.mqtt.client as mqtt
004.
005. # Change these to suit your needs
006. broker = '192.168.0.100'
007. client_name = 'busybot'
008. topic = 'study/busybot'
009. brightness = 0.1
010.
011. current_message = ''
012.
013.
014. def scroll_message():
015.     # Original function by Pimoroni x
016.     global current_message
017.     while True:
018.
019.         # No message? Don't do anything.
020.         if len(current_message) is 0:
021.             scrollphathd.clear()
022.             time.sleep(1)
023.             continue
024.
025.         # Clear the display and reset scrolling to (0, 0)
026.         scrollphathd.clear()
027.         length = scrollphathd.write_string(current_message)
028.         scrollphathd.show()
029.         time.sleep(0.5)
030.
031.         length -= scrollphathd.width
032.
033.         # Now for the scrolling loop...
034.         while length > 0:
035.             # Scroll the buffer one place to the left
036.             scrollphathd.scroll(1)
037.             scrollphathd.show()
038.             length -= 1
039.             time.sleep(0.02)
040.
041.         time.sleep(0.5)
042.
043.
044. def on_message(client, userdata, message):
045.     global current_message
046.     print('MQTT Message received')
047.     current_message = message.payload.decode("utf-8")
048.     if len(current_message) == 0:
049.         print('Clearing message')
050.     else:
051.         print('Scrolling ' + current_message)
052.
053.
054. print('Starting')
055. client = mqtt.Client(client_name)
056. client.connect(broker)
057. client.subscribe(topic)
058. client.on_message = on_message
059.
060. print('Listening to ' + topic)
061. client.loop_start()
062.
063. scrollphathd.set_brightness(brightness)
064. scroll_message()
```

FreeCAD: technical drawings

In this fifth part of the FreeCAD series, we are going to explore the TechDraw workbench – allowing us to create excellent technical drawings of our FreeCAD projects

Jo Hinchliffe

@concreted0g

Jo Hinchliffe is a constant tinkerer and is passionate about all things DIY. He loves designing and scratch-building both model and high-power rockets, and releases the designs and components as open-source. He also has a shed full of lathes and milling machines and CNC kit!

Many users will be using FreeCAD to model in 3D to produce files that can be sliced and printed on a 3D printer. Other users, however, will be perhaps designing parts that they want to create using subtractive methods, like lathes or milling machines. Some will be using bench-fitting techniques to cut, drill, and file to hew parts from materials. To share the information needed to manufacture a part, the standard practice is to use technical drawings. Drafting a technical drawing using pencil and paper and a drawing board is a fantastic skill to have, but it can be difficult to master. FreeCAD (**Figure 1**) simplifies the creation of technical drawings from 3D models by having a dedicated technical drawing workbench called 'TechDraw'.

To work through an example of creating a technical drawing, we need a 3D model to use. Let's quickly draw an item using simple techniques we learnt in the first parts of this series, starting in issue 37, using the Part Design workbench.

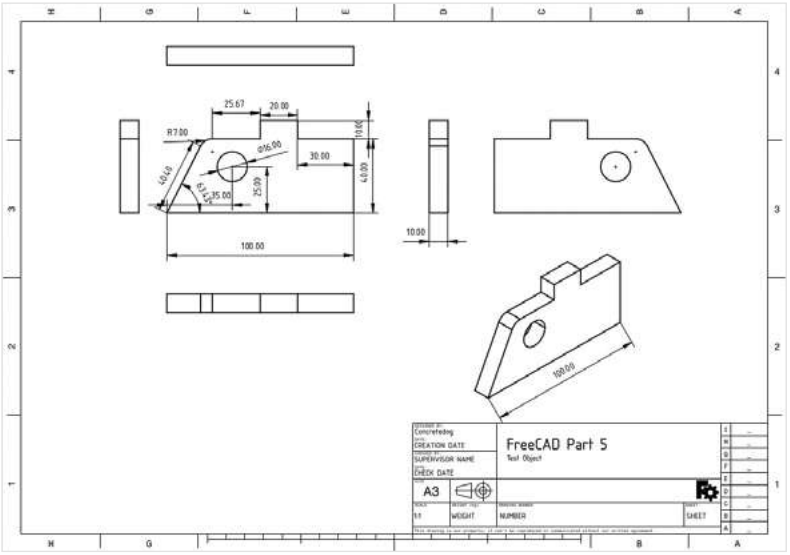


Figure 1 FreeCAD allows the easy creation of technical drawings of our 3D parts

Create a new project and, on the Part Design workbench, create a body and then a sketch in the XY plane. Using the 'Create Polyline' tool, create a shape similar to **Figure 2**. We don't necessarily need to constrain the entire model or sketch to use the TechDraw tools, but it's useful in this case to have all the dimensions constrained so we can compare the dimensions on the drawing (more on why that's important later).

Having drawn the outline, add a circle in the object, and then close the sketch and perform a pad operation on the sketch to create our part. Finally, before we jump into the TechDraw workbench, add a fillet to an external edge with a 7 mm radius. Having an internal hole and an external radius and an angled edge gives us enough details to do a good demonstration of the TechDraw workbench tools – see **Figure 3**.

To begin playing with TechDraw, select the TechDraw workbench from the drop-down menu. Click the 'Insert Default Page' tool icon and you should have a new tab in the preview window with an empty technical drawing page in it – **Figure 4**, overleaf.

You can see that the new 'page' is listed as an object in the file tree. Switch back to the tab that contains the 3D model and you'll notice that, despite switching tabs, you are still in the TechDraw workbench. As an experiment, move the model so that it is at an angle rather than in one of the 'front, top, back' type views. Next, ensuring the body is highlighted in the file tree, click the 'Insert View' tool icon. If you switch to the 'page' tab, you should see the 3D object has been added to the drawing in its exact view, at the angle it appears on the live preview. It's important you know that it does this, because if your object is slightly at an angle,

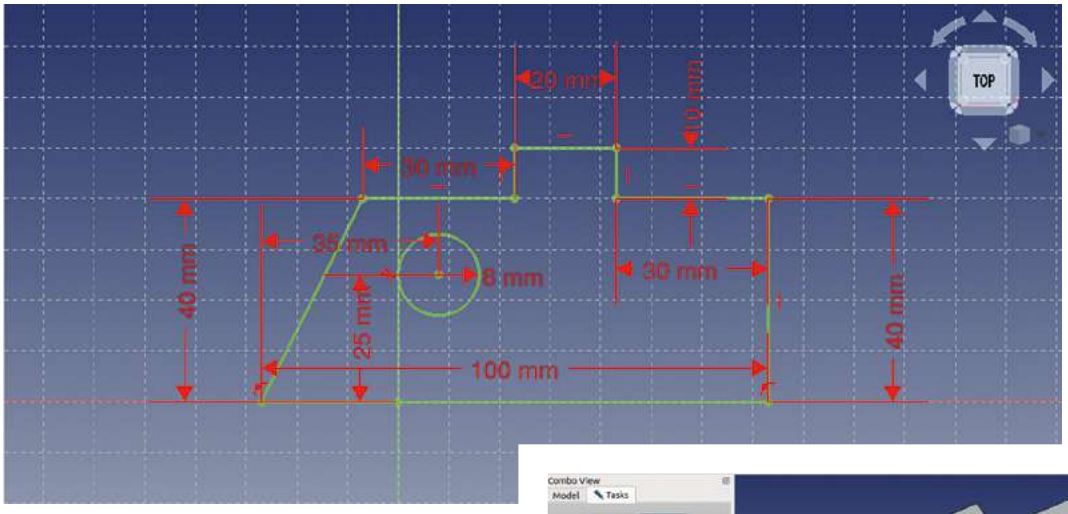


Figure 2
The basic sketch of our test object to explore the TechDraw workbench

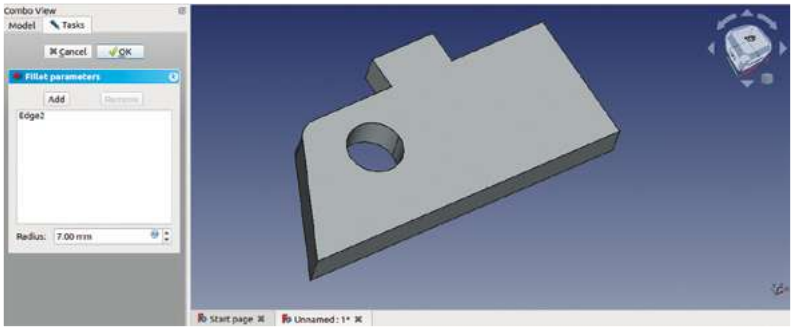


Figure 3
Our object, now padded and with a 7 mm radius fillet added

you might not notice, but it can affect dimensions in technical drawings added later. On the 'page' tab, you should have a dashed line around the 'view' – these are referred to as the 'View Frames'. Clicking the frame and dragging allows you to move the view around the technical drawing page. Highlighting the frame and clicking delete allows you to remove it. Click and delete this angled view and let's have another go. Return to the preview window and, using either the view icons, or the rotation block, select the 'top' view. Repeat the process of clicking the 'Insert View' tool to add the view to the page,

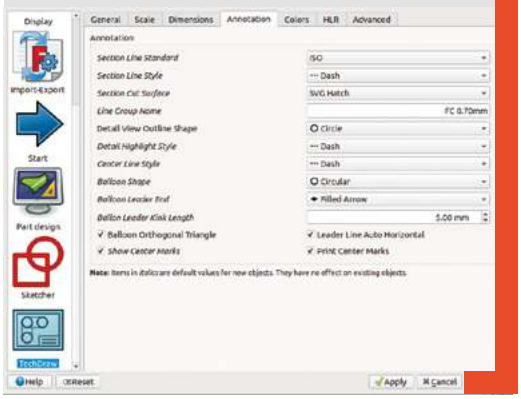
> **Our simple test piece design doesn't need masses of views** to be able to show all of its attributes and dimensions

(**Figure 5**, overleaf). You'll notice that all the lines in the drawing have the vertices marked as dots (similar to the Sketcher layout). The vertices help us to add dimensions and other tasks, but they are part of the view frame. You can toggle the view frames on and off by clicking the 'Turn View Frames On/Off' icon, or by right-clicking and selecting 'Toggle Frames'.

Our simple test piece design doesn't need masses of views to be able to show all of its attributes and dimensions, but we would definitely need another view adding to show the thickness of the part. Later, we will look at automatically generating multiple views of a part which adhere to technical drawing →

GETTING SET UP

Before we add any views to our new page, we need to make an important change in some preferences so that the centres of circles and arcs are marked when they are drawn in TechDraw views. Making sure you are on the TechDraw workbench, go to Edit > Preferences and then scroll down the left-hand side to the TechDraw icon. On the TechDraw preferences window, click the 'Annotation' tab and make sure the 'Show centre marks' and 'Print centre marks' boxes are checked. Click Apply, and then OK to close Preferences.



YOU'LL NEED

- A laptop or desktop computer

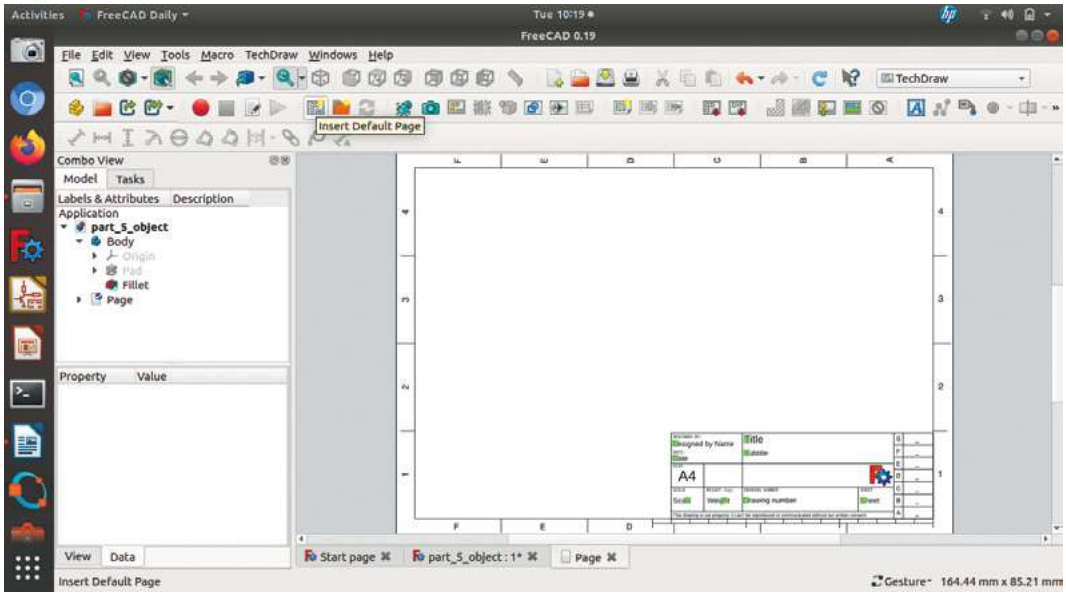


Figure 4
Starting a first technical drawing by inserting a blank page

ANNOTATIONS

Most technical drawings will have some details added as text; in our default template, we have some text boxes set up in the lower right-hand corner. These include common details you might wish to add to a technical drawing, including the title, designer name, date, scale of the drawing, and more. With the view frames turned on, you should notice that each text area has a small green block. Clicking on the green block allows you to edit and insert the text that you want to add. You'll also notice that there are some blank boxes with no text inserted. You can add text anywhere on the document by using the 'Insert Annotation' tool. Clicking the tool, you should see a view frame appear labelled 'annotation' with 'default text' written in it. To edit the annotation, select the view frame either by clicking it in the preview, or selecting it in the file tree view. In the combo view, you should now see the editable parameters of the annotation, which include common text editing options like font and size. At the top of the annotation parameters, you should see 'Text' and the 'Default Text' that is stored within it; click the ellipsis icon (the three dots to the right) and in the pop-up window you can input text. Note that it doesn't use text wrapping, and you need to click the **RETURN** key to create a new line of text. In the image above, we have filled in the default drawing details with the editable text boxes, but also added the lines about the part tolerance and surface finish by adding an annotation box. Notice we also added some text in an annotation box outside the default drawing text boxes, as you can add an annotation frame anywhere in the drawing.

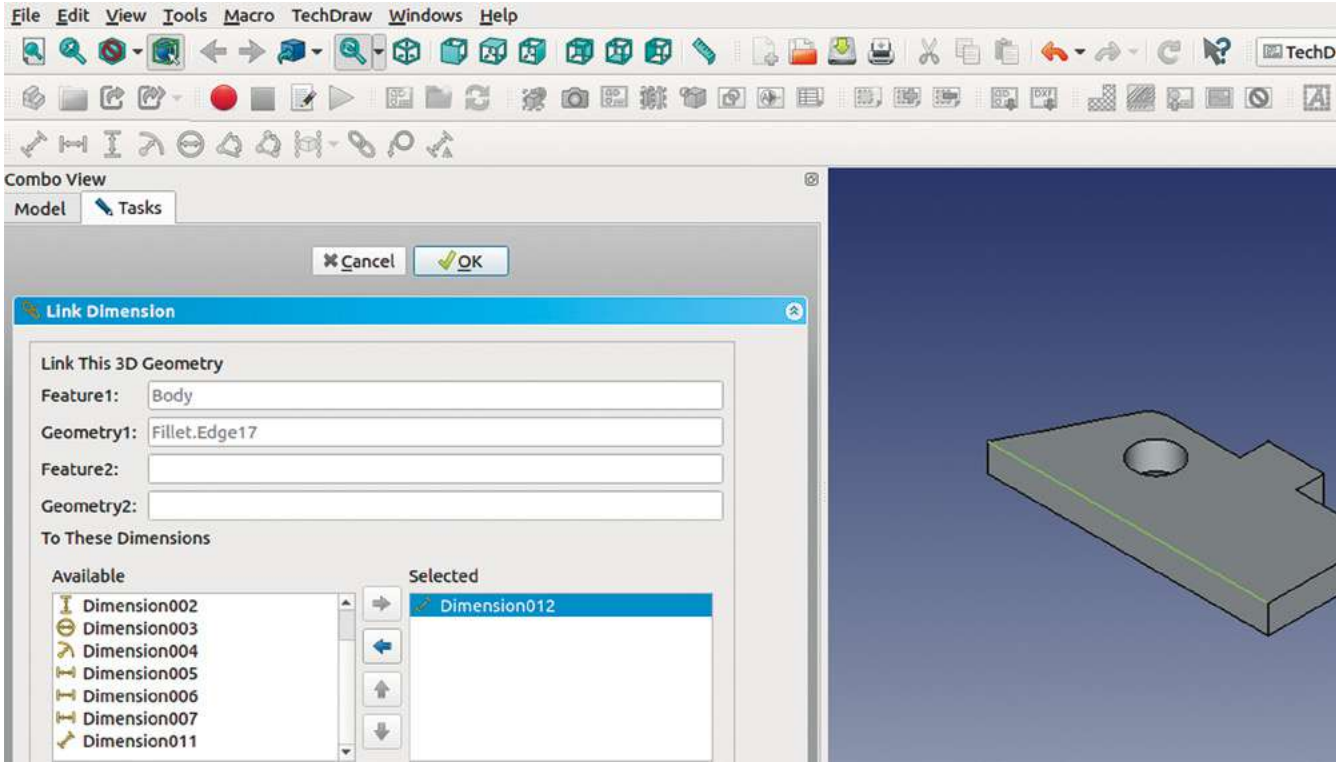
conventions, but for now, let's just add another view. Do this manually as we did with our top view. Return to the 3D model preview, and move the model so we are in a left- or right-hand view, then click the 'Insert View' button.

Next, let's add some dimensions to parts of the views so that someone could make this part accurately. It's similar in feel to adding constraints in the Sketcher workbench. First, let's select the long line/edge at the base of our object in the front view on the technical drawing page. Then click the yellow 'Insert Horizontal Dimension' tool icon. You should now see a dimension appear in the drawing. Similar to the Sketcher workbench, you can click and drag this dimension to different areas to organise where the labels are. Adding the height of the 3D object is a similar task, except we don't have a single complete line from the bottom to the top of the object, so we will use vertices instead. Similar to the vertical and horizontal constraints in the Sketcher workbench, TechDraw can create a vertical dimension between two points, even if they don't sit on the same edge.

> **You can click and drag this dimension to different** areas to organise where the labels are

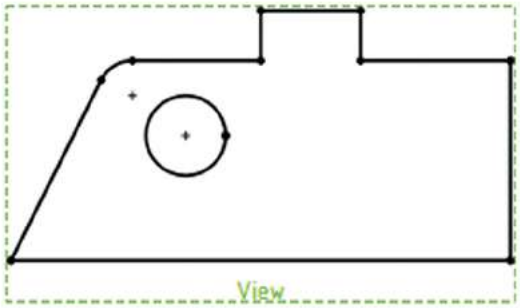
QUICK TIP

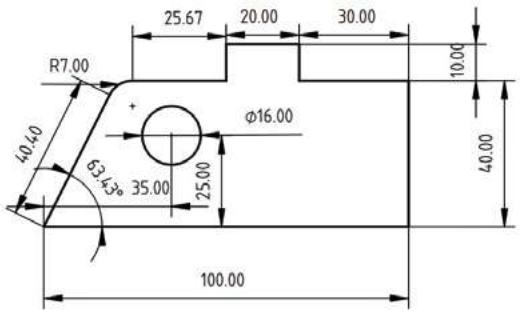
If a linked dimension doesn't update straight away, clicking and moving the dimension slightly often refreshes it.



Slightly differently than the Sketcher workbench, when you want to select multiple vertices, you need to hold the **CTRL** key down. Select the right-hand vertex on the bottom edge of our object and the highest vertex on the right of the small bit that sticks up at the top of the design. Click the 'Insert Vertical Dimension' tool to add the dimension and note that it correctly adds the vertical distance between the points – not the distance along a straight line between the points.

Holes in designs in technical drawings are commonly annotated with the diameter rather than the radius, often as the maker will be wanting to know what size drill or reamer to use. Adding a dimension for the diameter of our hole is similar to adding a radius constraint when we drew the circle in Sketcher. We simply select the circle and then click the yellow 'Insert Diameter Dimension' and the diameter of our hole is displayed. To indicate the co-ordinate position of the hole centre, we can select the centre mark and apply horizontal and vertical dimensions relative to another vertex. We have selected the lower left-hand corner of the object to act as this datum point and, of course, you could repeat this for the centre marks of the arc of the fillet. Most technical drawings would have a radius of the arc marked, rather than a diameter, and this would apply to our fillet arc. To add this, simply click the arc line and then click the 'Insert Radius Dimension' tool – **Figure 6**.





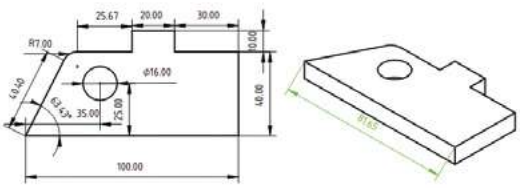


Figure 8
Linking a drawing dimension back to the model is useful in angled isometric views

Figure 5
Our first view of our object inserted into the new drawing

Figure 6
Our view of our object with most dimensions added

Figure 7
Adding a line length to an isometric view can create an incorrect dimension, as seen here

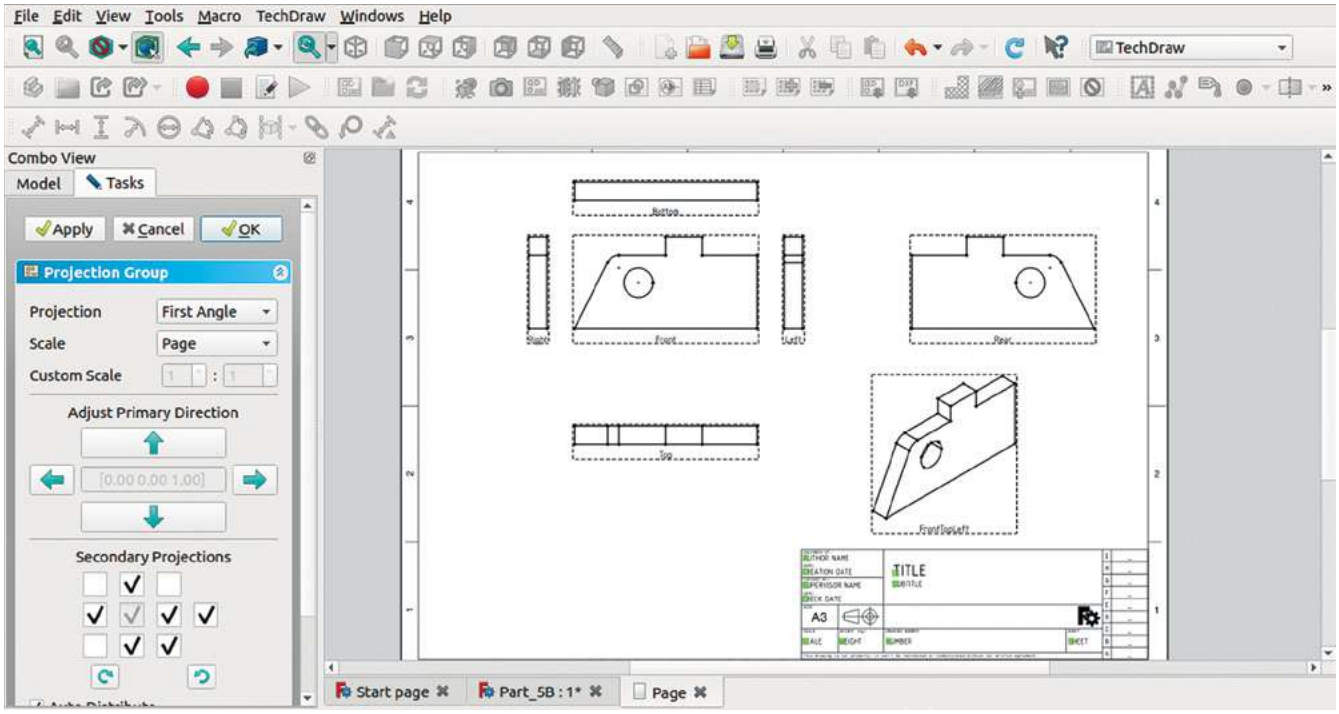


Figure 9 Selecting views to insert multiple views in one pass

To add a dimension showing the angle of the left-hand side line, relative to the base of the object, select both these lines and click the yellow 'Insert Angle Dimension' tool icon. Whilst it's not often necessary, sometimes we might want to add the actual length of a line rather than a vertical or horizontal and, to do this, let's select the angled line and click the yellow 'Insert Length Dimension' tool. This is a very useful function at times, but also it hints at the way TechDraw handles line/edge length

> **This is a very useful function at times, but also it hints at** the way TechDraw handles line/edge length

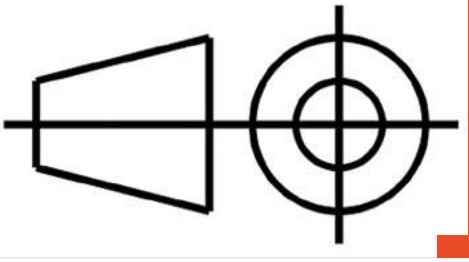
and a possible problem. If we return to our 3D object in the live preview, click the 'Set to Isometric View' icon from the collection of blue view icon tools. With the object now at an angled isometric view, let's add a new view to our TechDraw page. Move to the technical drawing page and move the new view so that it's clear of other views. Select one of the long lines that is the base of our object and click the 'Insert Length Dimension' tool. If you compare the dimension you have just added to the very first horizontal dimension we added on our first view of the object, you'll notice that they are different. This is because the TechDraw dimensions default to being

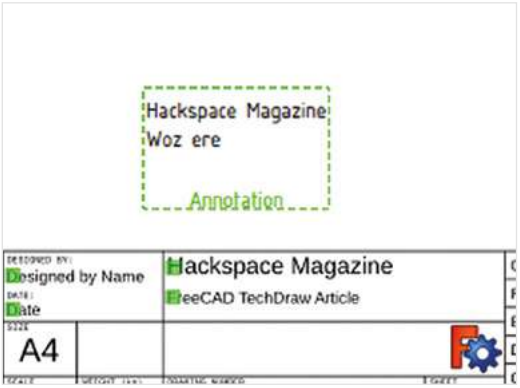
WHICH ANGLE?

Technical drawing in itself is a massive field to learn about, and there are numerous standards and systems in place governing technical drawings. One set of standards relate to the projection angles which affect where views are positioned in a technical drawing. These two projection views families, 'First Angle' and 'Third Angle', are both used but create a technical drawing laid out differently. First angle drawings are more common outside of North America, whereas in North America third angle is more common. It's important that you indicate that you are using a first or third angle projection on a technical drawing and that you only use that type of view and don't mix them. Commonly on technical drawings, you indicate whether a drawing is first or third angle by adding the correct icon, as seen in the image below.

It's quite complex to explain the difference, and there are numerous different approaches to explaining first and third angle.

A good place to start is this Wikipedia page: **hsmag.cc/Angle**.





based off the view drawing rather than the actual 3D object, so the actual length of the angled line in the isometric view drawing is shorter (**Figure 7**, overleaf).

If you wanted to add an isometric view with some dimensions added, you can overcome this problem by linking a dimension to the 3D object. First, highlight the incorrect dimension we just added to the isometric view, and look in the file tree view to identify the name of this dimension. In our example, this was 'Dimension012'. Having noted that, click over to the 3D object tab and highlight the same edge on the 3D object. Next, click the yellow 'Link Dimension to 3D Geometry' tool. In the combo view, you will see a window with a list of dimensions in a left-hand column. Select the dimension name that we noted earlier and then click the right-facing arrow to bring that dimension name over to the right-hand column – **Figure 8**, overleaf. Click OK at the top of the menu and return to the technical drawing page tab. You should now see that the dimension has updated to the correct length of the 3D object.

Having played with the TechDraw tools a little, let's delete our practice drawing and start again, looking at a couple of things that might make the process quicker now we know how some of the tools work. Select the page we were working on in the file tree and delete it. To create a new page for our technical drawing, let's click a different icon: the 'Insert Page Using Template' icon that looks like an orange folder. This opens a folder containing different templates for technical drawings. There are some things that will appear obvious to you, like there are different page sizes: A4, A3, A1, etc. There are also some templates that adhere to particular technical standards, such as the American National Standards Institute (ANSI) and the International Organisation for Standardisation (ISO), which may be of use if you want to make standard-compliant drawings. Let's use a page called 'A3_LandscapeTD.svg'.

Make sure that the 3D object is in the top view in the preview and highlight the body in the file tree. This time, instead of clicking to insert a single view, let's use the 'Insert Multiple Linked Views Of Drawable Objects' tool icon, which should be next to the template folder icon. If you now click to the 'page' tab, we can see any changes we make in the dialog box live on the template.

In the dialog box, the first thing you will see is a projection menu. Let's use first angle, as our template indicates the drawing will be in first angle projection. For this example, we don't need to scale the drawings as it's quite a small object. However, you can play with selecting a custom scale and changing the values, which is useful if you want to scale down a large object on the plans. Further down you will see an arrangement of checkboxes that correspond with the multiple views that will be added to our drawing. Select the pattern we have selected in **Figure 9** and click OK. On the technical drawing, you should now have the corresponding views automatically inserted. To move linked views is slightly different as it always aligns the projection of the main views to the centre view. To move all the views as a group, move the centre view; to move the upper, lower, and side views, you can push and pull them relative to the centre view. The exception is our isometric view, which we can move anywhere in the page.

As ever with FreeCAD, there's heaps more we could look at on this workbench alone, but hopefully you are well on the path to creating beautiful and useful technical drawings for all your project needs. ◻

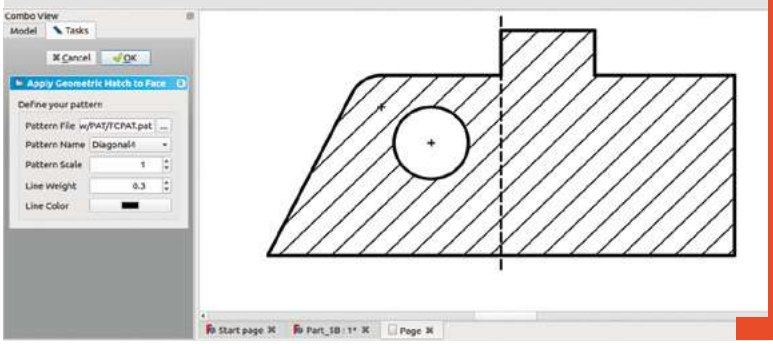
Left
Figure 10. Adding text can help your viewers understand what's going on

QUICK TIP

To export and print your technical drawing at any time, simply right-click anywhere in the drawing to get a range of output option.

GETTING CENTRED

Technical drawings often use centre lines and hatching and other effects to make them more readable. Adding a patterned centre line is possible, and you can use a face, two vertices, or two lines as a reference. As a quick example, click the face of our object in the front view to select it and then click the 'Insert Centre Line' tool icon. In the dialog box you can adjust the position and appearance, but for now just click OK and a dashed line should appear vertically through our object. A centre line is classed as a 'cosmetic object' and as such doesn't appear in the file tree. To remove the centre line we just created, you need to select the centre line and then click the 'Remove Cosmetic Object' icon, which looks like an eraser. To add hatching to a face, select the face and then click the 'Apply Geometric Hatch To Face'. In the dialog you can change various settings such as the pattern, line weight, and colour.



Connected Little Boxes

Discover how easy it is to create a network of Internet of Things devices with no programming required

Rob Miles

@robmiles

Rob Miles has been playing with hardware and software since almost before there was hardware and software. You can find out more about his so-called life at **robmiles.com**.

Figure 1 Some connected little boxes

The Internet of Things (IoT) is the future, so they say. But what if you don't want to write much program code? Read on to discover how to build a range of connected devices controlled by ESP8266 or ESP32 processors that can be managed remotely and do all kinds of wonderful things. The devices are controlled by simple commands and have a wide range of configuration options. Everything is controlled by little packets of JSON data. And if you're not sure what JSON (JavaScript Object Notation) is, you'll learn that too.

CONNECTED LITTLE BOXES

The 'Connected Little Boxes' project grew from something simple: a desire for a light in one house to change colour when someone walks in front of a sensor in another house. The idea was to create a gentle way of keeping in touch. You wouldn't know what the other person was doing, but at least you'd know they were up and about. Then the project put on a bit of weight as other devices were added, including buttons, servos, rotary encoders, printers, environmental sensors, and text displays.

Figure 1 shows some of the Connected Little Box designs. Some have just lights in them, some have buttons and passive infrared (PIR) sensors, and some have rotary encoders and servo outputs. They can send messages to each other, and they can also be controlled by a remote server application.

You don't have to build all the boxes. You can start with just one box you can control from the internet and then add others later. All the software, circuit, and box designs are available on GitHub at the Connected Little Boxes organisation site: **hsmag.cc/CLB**. The **box-code** repository contains the code for the boxes, **box-hardware** the hardware designs, and **box-server** the code for the Node.js server you can use for remote control and management.

Each connected device runs the same software and is individually configured for its role. All the input and output signals can be directed to different physical pins in the hardware for that device.

OUR FIRST CONNECTED LITTLE BOX

In **Figure 2**, you can see a box that contains a push-button and a pixel ring. This box is controlled by a WEMOS D1 mini, which is an ESP8266-based device.

You can add things to a Connected Little Box, as shown in the circuit diagram (**Figure 3**). These are the suggested connections; you can use different pins if you wish and modify the device settings accordingly.

Figure 2
A connected little box with lights and a button

Figure 3
Box circuit diagram

All the elements can be controlled individually. You could make the servo move or the light change colour (or both) when the PIR sensor detects movement or when the button is pressed. A box can send commands to other boxes so that you can move the servo on another box when the button on this box is pressed. It will also respond to incoming messages too. If you have an ESP8266 device with NeoPixels connected to it, as shown in **Figure 3**, you can create a remotely controlled light. Let's see how you would do that.

GETTING THE SOFTWARE

You can get started with Connected Little Boxes by building the software and downloading it to an ESP32 or ESP8266. The software is all written in C++, and the best way to compile and deploy it is to use Visual Studio Code, which is available for Raspberry Pi. You will need to add the PlatformIO plug-in to Visual Studio Code.

Figure 4 (overleaf) shows the Connected Little Boxes project open in Visual Studio Code. It looks complicated to use, but there are only two buttons you need at the moment. Connect an ESP8266 device to your computer via USB and press the Deploy button. This will build the software and then deploy it over USB into your device. It may take a while to do this. Once the software has been deployed, you connect to your new Connected Little Box and have a chat with it.

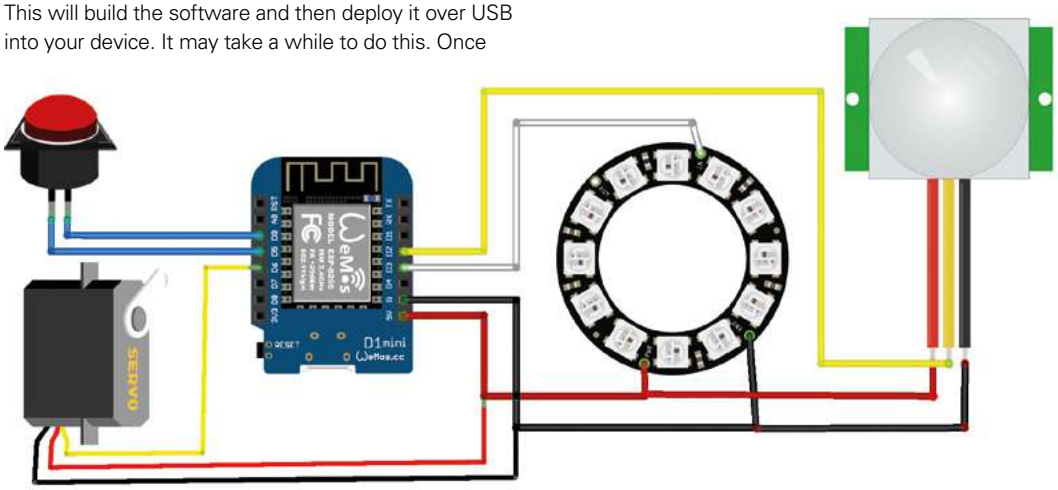
CHATTING WITH A CONNECTED LITTLE BOX

A Connected Little Box provides a set of commands that you can use to configure and manage it. All the settings that you enter are retained when the box is switched off. To connect Visual Studio Code to your box, press the Terminal button indicated in **Figure 4**. Your dialogue with the box will take place in the terminal window, which is on the bottom right-hand side of the Visual Studio Code window. When a box starts up, it starts up all the processes (things that control devices) and sensors (things that provide data and triggers). At the end of all this, the box will display a welcome message and then wait for you to type in a command.

```
Start complete

Type help and press enter for help
```

Click inside the terminal window so that the cursor in the window turns solid (fully black or fully white depending on the selected Visual Studio Code colour scheme), and then type in your command. If you type 'help' and press the **ENTER** key, the box displays a list of all the commands that it understands. At the moment we don't want to give the box commands, we want to configure some settings. We can do this by typing in commands that assign values to the settings that we want to change:

```
wifissid1=yourWifiSSID
Processing: wifissid1=ZyXEL56E8A7 value set
successfully
wifipwd1=yourwifipassword
```



QUICK TIP

Use a spare GPIO line as ground. When you are connecting inputs to your microcontroller, you often run out of connections for the ground line because, like the WEMOS in **Figure 3**, most microcontrollers only have one ground pin. If you have a spare input/output pin, you can use this as a ground connection. Just remember to set the 'ground' pin as an output, and set the level to low.

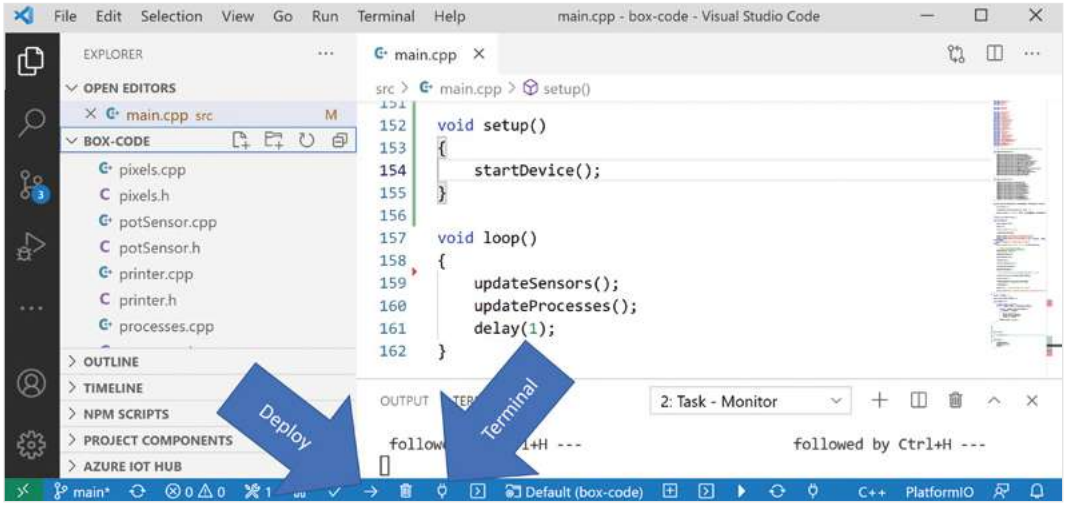


Figure 4 Building the software in Visual Studio Code

YOU CAN USE THIS WITH

- **A device you want to control with an ESP8266 or ESP32 processor**
- **A local MQTT server such as Raspberry Pi running the Mosquitto broker software, or an internet-based server such as HiveMQ.** You can download Mosquitto from: mosquitto.org and find HiveMQ at hivemq.com/public-mqtt-broker
- **Coloured pixels, buttons, PIR detectors, LED arrays, potentiometers, shaft encoders, printers, and anything else you fancy connecting**
- **A 3D printer. Printable designs are available to make cases for all the boxes**
- **A Node.js server such as Raspberry Pi, which can host the web-based remote management interface** (this is not required to use the boxes)

```
Processing: wifipwd1=yourwifipassword value set
successfully
```

CONFIGURING MQTT

Now that we have our box on a network, we need to configure it so that it can communicate with other boxes. Communication between Connected Little Boxes is performed using the Message Queue Telemetry Transport (MQTT) protocol. Boxes can post messages to addresses (called 'topics') on the broker and subscribe to topics to receive messages from other boxes. You can set up a broker of your own, or you can use an open one. To get started, we can use the open broker at **hivemq.com**:

```
mqtthost=broker.hivemq.com
```

The command above configures a Connected Little Box to use the open broker at **hivemq.com**. If the box has a working network connection, it will connect to the broker.

```
OK: 2 Mqtt OK
Publishing {"name":"CLB-
aa60a4","processor":"ESP8266"
,"friendlyName":"","version":"1.0.0.11",
"processes":  [
```

GET CONNECTED

When you have entered your WiFi credentials, you can restart the box by using the restart command:

```
restart
```

The box will go through the restart procedure and then tell you that it is now connected to the WiFi and display its IP address.

```
OK: 1 WiFi connected OK 192.168.4.192
```

```
"pixels","console","controller"],"sensors": [
"clock"]} to registration
OK: 21 Mqtt transmit OK
Publishing {"name":"CLB-aa60a4","reset":"External
system reset","cpu":"ESP8266","resetcode":6} to
connected
OK: 21 Mqtt transmit OK
```

When a box connects to MQTT, it publishes two messages to say 'hello'. These include the reason it was restarted, the MQTT device name, and the sensors and processes that are active on that device. These messages are used by the server application to manage the device. One of the principles of MQTT is that messages are only received by those who are listening for them, so unless there are devices out there listening on the registration and connected topics, the messages will be ignored.

TALKING TO A BOX OVER MQTT

Now that our box is on the network and connected to an MQTT broker, we can send it a message. However, before we do this, we need to know the address of our box. Each box has unique addresses which are assigned when the box is first started. We need to know what these are. We can find out everything that a Connected Little Box knows about MQTT by using the **dump** command to dump the MQTT setting values. The **dump** command can be followed by a filter so that only settings with names that contain the filter string are displayed.

```
dump mqtt
```

The command above will ask a box to display only its MQTT settings.

```
mqttdevicename=CLB-aa60a4
mqttactive=yes
mqtthost=broker.hivemq.com
mqttport=1883
mqttsecure=no
```

```
mqttuser=
mqttpwd=******
mqttpub=data/CLB-aa60a4
mqttsub=command/CLB-aa60a4
mqttreport=report/CLB-aa60a4
mqttsecsperupdate=360
mqttsecsperretry=10
```

The settings that we are interested in are the **mqttpub** and **mqttsub** ones. The **mqttpub** setting is the topic that the device will publish to when it sends a message. The **mqttsub** setting is the topic that the device subscribes to. Let's enter these into the HiveMQ MQTT client and send some commands to our box. You can find the client at **hsmag.cc/HiveMQ**. If you navigate to this page, you can then press the Connect button to open the connection.

Figure 5 shows how to send a message to a box. The Publish topic must be set to the **mqttsub** setting in the Connected Little Box. The Add New Topic Submission button lets you specify a topic for the web page to listen to. This should correspond to the **mqttpub** setting in the Connected Little Box. When the Publish button is clicked, the contents of the message text box are sent over MQTT to the specified topic. Because the Connected Little Box has subscribed to the topic, it will receive the message.

```
{"process":"pixels",
"command":"setnamedcolour","colourname":"green"}
```

This is the command that is sent in **Figure 5**. It sends the command **setnamedcolour** to the process **pixels**. The named colour is specified as 'green'. The result is that the LEDs on the target box will turn green. The terminal output of the Connected Little Box will display a logging message indicating that the message was successfully received.

```
Received from MQTT: {"process":"pixels",
"command":"setnamedcolour","colourname":"green"}
Publishing {"error":0} to data/CLB-aa60a4
OK: 21 Mqtt transmit OK
```

When a message is received, a Connected Little Box will respond with an error code. An error code of '0' means that everything works. You can see that message in the Messages area on **Figure 5**.

CONNECTED LITTLE BOX COMMANDS

You have seen your first Connected Little Box command. A command is a tiny piece of JSON text. JSON is used throughout the World Wide Web to pass messages between processes. Items in a JSON-formatted message can have names and values. Names are enclosed in double quotation marks, and values can be strings of text in double quotes or a numeric value. Each item in a JSON message is separated from the next by a comma, and the whole message is enclosed in curly brackets.

```
{"process":"pixels","command":
"brightness","value":0.1}
```

Above is another command that you can send to your Connected Little Box. This one sets the pixel brightness to 0.1. The maximum brightness level is 1, so this →

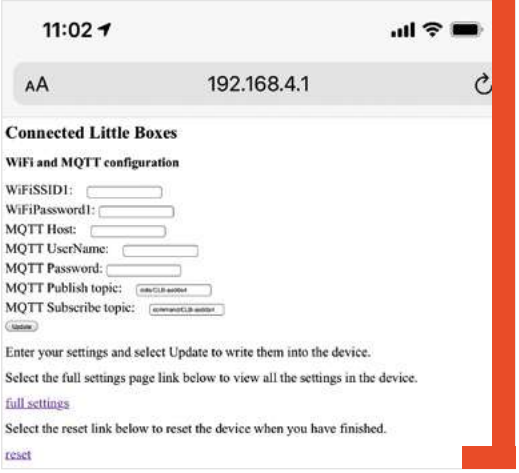
BROWSER- POWERED CONFIGURATION

You can also configure a Connected Little Box using the browser on your mobile phone. If the box starts running and discovers that it has no WiFi settings, or it takes longer than five minutes to make a connection, it will start a WiFi access point and web server that you can connect to and browse to enter the settings information.

CLB Access Point

CLB Web Address

A good way to configure a connection to a device is via QR codes, which you can generate easily on the internet. The user can just use the camera app on their phone to view the tags and open the access point and configuration website. The website **qifi.org** will create WiFi QR codes, and the website **the-qrcode-generator.com** will make website codes. You can then print these onto labels and stick them on the device for easy configuration.



Left QR codes can configure your WiFi and contain web addresses

Left Configuring a box with your phone

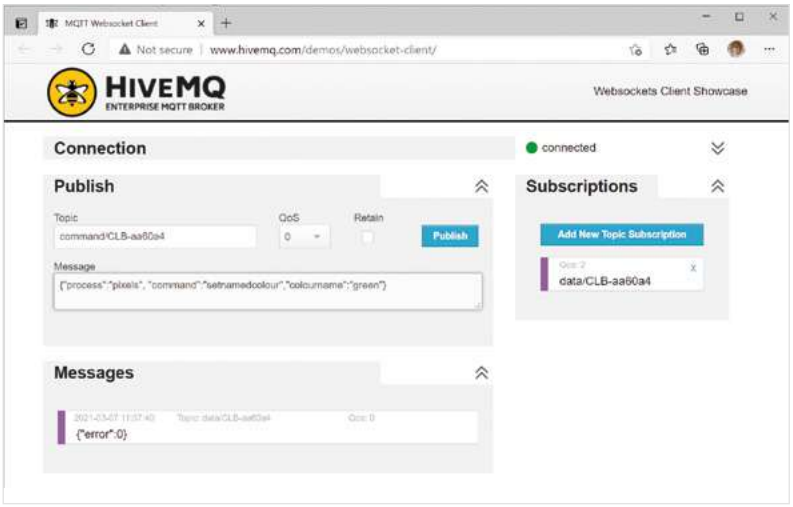


Figure 5 Talking to a box from the web

Below Design for the box with a button and an LED ring

makes the pixels quite dim. Commands can also be entered via the terminal, so if you type this command into the Visual Studio Code terminal window connected to a box, you should see the pixel brightness change. You can ask the Connected Little Box what commands it knows by using the following terminal command:

```
commands
```

You might regret asking the question, though, as the box will produce quite a lot of output, and none of it looks easy to understand. However, if you look carefully through it, you will find that it is actually JSON that contains a description of each of the commands and what they act on. Each item has a text description, whether the item is optional (1 means the item can be omitted), and a description.

```
commands
{"name":"brightness","version":"
1.0.0.11","desc":"Sets the pixel brightness",
"items":[{"name":"value","optional":0,
"desc":"value (0-1)","type":"float"},
            {"name":"steps","optional":1,
              "desc":"no of 50Hz steps to
complete the change","type":"int"}]}
```

Above is the description of the brightness command that we have just used. With a little detective work, you should be able to work out that the description above applies to the pixel brightness command, and that this command also accepts an option called **steps**, which can be used to control how fast the pixels fade to the specified brightness. You can use this to make lights that fade down slowly.

```
{"process":"pixels","command":
"brightness","value":0,"steps":500}
```

The statement above would perform a long, slow fade. You may be wondering why the box is making it so hard for you to understand what it does; this is because this response is intended for use by another computer system that might want to know what the box can do.

USING SENSORS

If you want to make a Connected Little Box that can respond to button presses, you can connect a 'push to make' switch between two button input pins. On the circuit diagram in **Figure 3**, you connect the switch between pins D5 and D0 of the WEMOS device. Unfortunately, the numbering of the pins on the physical ESP8266 processor doesn't match up with the pin numbers on the WEMOS board. As far as the ESP8266 is concerned, these two pins are actually numbered 14 and 16, and these are the initial settings for these pins in a new box.

This means that if you connect a button as shown in the circuit diagram, it'll work, but don't be confused by the strange numbers you see in the settings. When you use any device with numbered pins, it's important to find out what the numbers refer to.

If you want to use a sensor, you must first enable it. You can enable the button sensor by turning it on with the following setting command:

```
pushbuttonfitted=yes
```

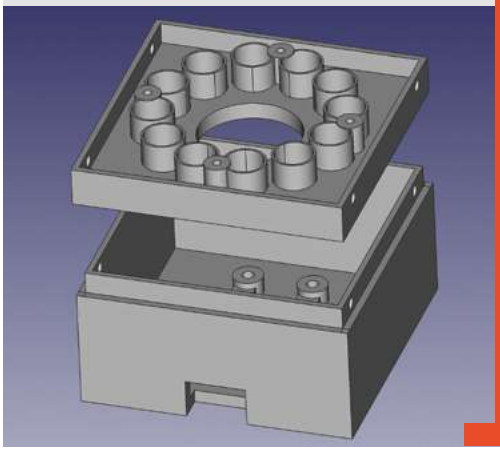
Now the Connected Little Box will react to the button. We can make it perform an action when

QUICK TIP

Using an ESP32 device. You can use an ESP32 device instead of an ESP8266. To change device, you use the PlatformIO project editor dialog. Open the PlatformIO page in Visual Studio Code, and then select PIO Home > Projects & Configuration from the Quick Access. Select the GitHub\box-code project, click the ESP32 DOIT tab, and select Make Default followed by Save.

BOXING CLEVER

There are 3D-printable box designs for lots of different boxes. The designs contain fittings to hold a WEMOS processor and provide external access to the mini USB connection. There are also programs that you can use to create boxes that contain your name in lights, as shown in **Figure 1**. You can find them on the Connected Little Boxes organisation site: **hsmag.cc/CLB**.



the button is pressed by adding a **sensor** element to a command:

```
{"process":"pixels",
"command":"setnamedcolour","colourname":
"green","sensor":"button", "trigger":"pressed"}
```

If you enter the command above into the Visual Studio terminal (or send it to a Connected Little Box via MQTT), you won't see the light change colour. However, if you press the button on the Connected Little Box, the light should turn green. Note that the light will stay green after you have released the button. If you want the light to turn red when the button is released, you'll need to add another command that is triggered when the button is released.

```
{"process":"pixels", "command":"setnamedcolour",
"colourname":"red","sensor":"button",
"trigger":"released"}
```

This **trigger** is fired when the button is released. Each **trigger** event is connected to a **listener** inside a Connected Little Box. You can find out what listeners are in use by using the command **listeners**:

```
listeners
Processing: listeners
Sensor Listeners

Process:pixels Command:setnamedcolour Sensor:button
Trigger:pressed
Process:pixels Command:setnamedcolour Sensor:button
Trigger:released
```

You add multiple listeners to a single trigger, and a box can have up to ten active listeners. You can clear all the listeners in a box using the command **clearlisteners**. The listener settings are retained when the box is powered off.

BOX-TO-BOX COMMUNICATIONS

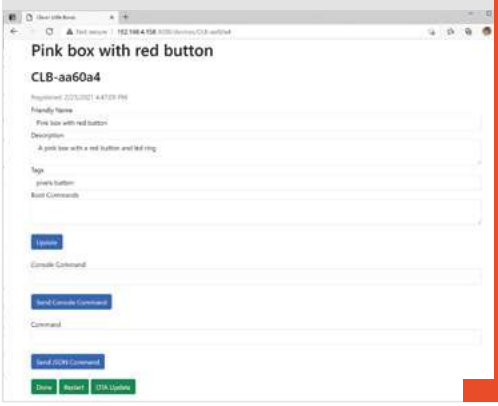
We can send commands to a box using MQTT, and we can configure sensors to trigger commands. Now we need to discover how one box can send a message to another. To do this, you just have to add a **to** element to a command which specifies the MQTT topic that's the destination for the command.

```
{"process":"pixels", "command":"setnamedcolour",
"colourname":"green","to":"command/CLB-eab714"}
```

The above command will turn the pixel green on the Connected Little Box with the address 'command/CLB-eab714'. By combining the **sensor** and **to** elements in a command, we can use sensors on one device to trigger actions on another.

WEB MANAGEMENT

The project also includes a web management tool that can listen for connection messages from boxes and build a database of devices. You can find the server code on the Connected Little Boxes organisation site: **hsmag.cc/CLB**.



Below
Managing a box over the web

```
{"process":"pixels", "command":"setnamedcolour",
"colourname":"red","to":"command/CLB-
eab714","sensor":"button", "trigger":"pressed"}
```

When the button on the local box is pressed, the light on the distant box will turn red.

TOWARDS AMBIENT COMPUTING

Computers that are embedded in the world around us that observe what we do, and respond automatically to our needs, deliver what we can call 'ambient computing'. If it sounds scary, it doesn't have to be. The behaviour could be something as simple as a device that turns on the lights when you enter a room. A Connected Little Box with a PIR sensor can do this – which means that the original aim of the project has been met. So, Connected Little Boxes are a tiny step on the road towards ambient computing.

The software inside a Connected Little Box has been designed to be extensible so that it is easy to add sensors and processes. There is comprehensive timing support via the clock sensor in every box, which automatically synchronises with internet time, and can trigger events on alarms and timers, as well as at regular intervals. There are connected little printers, text displays, servos, potentiometers, environmental sensors, and rotary encoders. The author is working on a connected little coin reader, which you can trigger by dropping coins into it. He's hoping that this will make him rich, since nothing else has worked so far.

QUICK TIP

Open is insecure. Only use open MQTT brokers if you really don't care about security. Anyone can send messages to a device that is connected to an open broker. You can improve security by using a broker that requires a username and password, or by using one that uses secure sockets for communications.

HackSpace
TECHNOLOGY IN YOUR HANDS

FIELD TEST

HACK | MAKE | BUILD | CREATE

Hacker gear poked, prodded, taken apart, and investigated

ONLY THE BEST

Electrifying your garden

Beginner-friendly kits and components for an aspiring digital gardener

By **Marc de Vinck** @devinck

Smart cars, smart homes, automation, and intelligent everything is all the rage both in the consumer market and with DIY enthusiasts, too. So why not add to the collective intelligence and take your plants online as well? Or at least give them a voice when it comes to their wants and needs. Just don't get out of control and give them a social media account. Although, that exists too!

In this Best of Breed, I'll be looking at some basic kits that will get you started in automating your gardening. Mostly monitoring the soil moisture level, but some will also allow you to monitor and control watering pumps or other environmental sensors. It's a collection of beginner-friendly, mostly no soldering required, components and kits that will help you get started in automating your gardening and getting your plants on their way to joining the digital world. Once you learn the basics of how to monitor the soil for moisture, you can start to think about other sensors like light, humidity, pH, and more. Automating an indoor garden is high on my list of things to do this year, and I hope it's on your list too!

Grow HAT Mini vs Grow Moisture Sensors

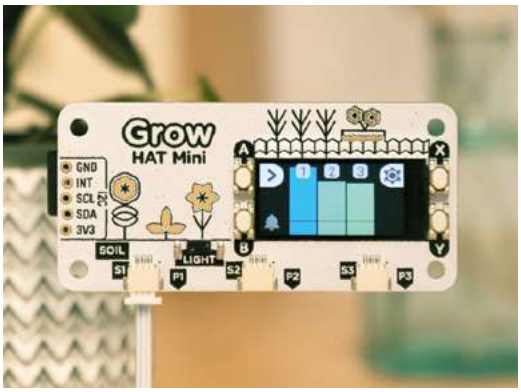
PIMORONI £19.50 | pimoroni.com

PIMORONI £10.80 | pimoroni.com

New from Pimoroni is the Grow HAT Mini for Raspberry Pi. This HAT is designed to help you monitor and take care of your houseplants. It can monitor multiple plants, notify you when they need watering, and even automate some of the other requirements of taking care of your plants when you are ready to expand the system.

The Grow HAT Mini features a tiny colour screen, which is a welcome addition when it comes to visualising the moisture levels, or other environmental conditions, of your plants. There's an integrated buzzer for when your plants need your attention, a light sensor for understanding when it's night-time and to keep quiet, and headers for connecting their Grow Moisture Sensors.

On the back of the board are connections for adding motors, valves, or other accessories. There's also a set of I2C connections, making the addition of other sensors simple. The Grow HAT is compatible with any flavour of Raspberry Pi, but my favourite is the Raspberry Pi Zero since it's a perfect mate when it comes to matching the Grow HAT's size. Check out Pimoroni's website for more information and a great 'getting started with the Grow HAT' guide.



Left
Turn your Raspberry Pi into a plant monitor

Below
Expand your Grow HAT with moisture sensing

The natural accessories to the Grow HAT Mini are the Grow Moisture Sensors from Pimoroni. The kit includes a trio of moisture sensors that are compatible with the Grow HAT and almost any microcontroller or single-board computer.

The sensors use capacitive moisture sensing with pulse frequency, making them a lot less likely to deteriorate over time, which can happen with other less sophisticated moisture sensors. The PCB has a handy indicator line to show you how far to push them into the soil, and there is even a space for naming or identifying your plants. If you are planning on picking up a Grow HAT, you definitely need to consider this three-pack of sensors to complete your smart water-monitoring system. →

VERDICT

Grow HAT Mini

A great place to start.

10/10

Grow Moisture Sensors

The perfect addition if you have a HAT Mini.

10/10

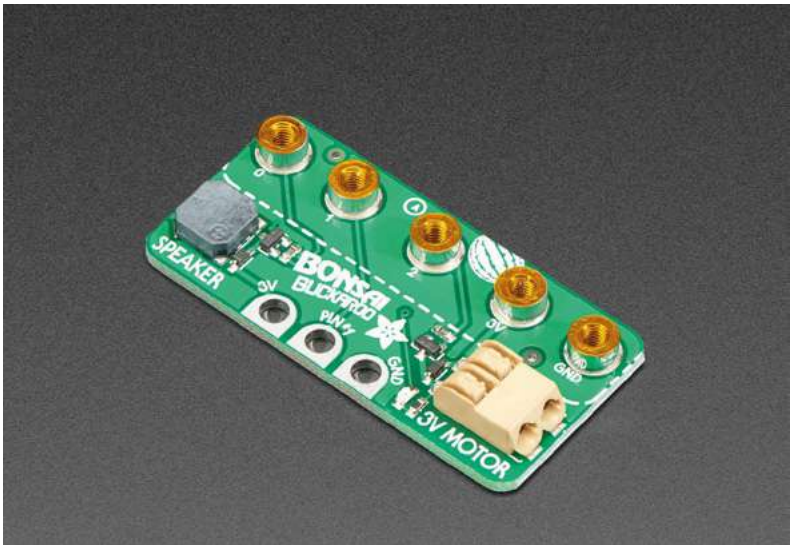
BEST OF BREED

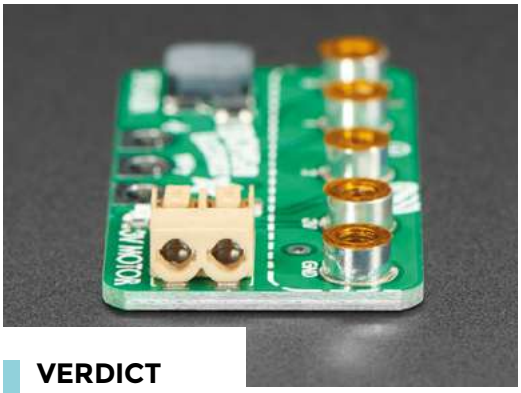
Adafruit Bonsai Buckaroo

ADAFRUIT $4.95 | adafruit.com

At the very least, the Adafruit Bonsai Buckaroo plant care helper board wins the prize for the best name in this roundup. Designed by Adafruit for the micro:bit or CLUE microcontroller platform, the Bonsai Buckaroo makes it incredibly easy to connect up a moisture sensor, water pump, and alarm, all without the need to solder.

The board features an 8mm buzzer for alerting you to low moisture levels, a connection point for a 3V motor to pump water as needed, and alligator clip pads for connecting up a moisture sensor. What kind of sensor? An alligator clip and a couple of nails will work just fine! Simple and no need to solder. This is a must-have accessory for anyone interested in giving their plants some intelligence, since it's so inexpensive and easy to use.





VERDICT

Adafruit Bonsai Buckaroo

A handy accessory for micro:bit gardeners.

10/10

Above
A quick and easy micro:bit plant care accessory

Red Hat Co.Lab Farm Kit

SPARKFUN ELECTRONICS ◈ $13.95 | sparkfun.com

Get ready to start your own open-source farm with the Red Hat Co.Lab Farm Kit from SparkFun Electronics. The kit includes everything you need to perform three different activities, all focused on basic electronics and growing microgreens.

You'll start by learning about breadboards by building a simple circuit to light up an LED. Next, you'll add a sensor for detecting moisture in your plant media. Then you'll connect it all together by lighting an LED when the growing media is too dry for your microgreens.

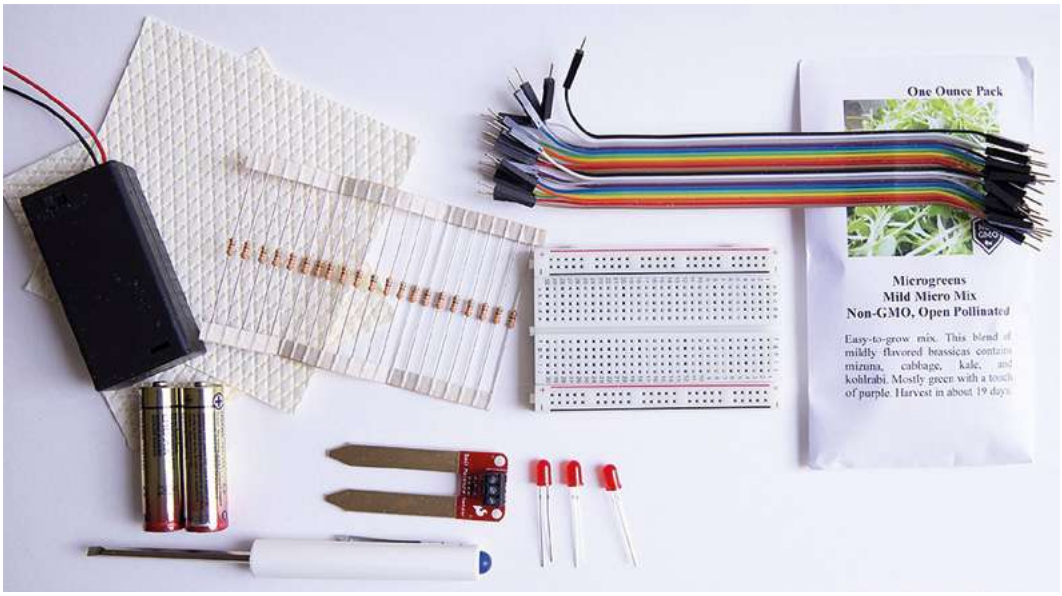
After you build the three different circuits, you should check out Red Hat's short documentary film, *Farming for the Future*, which shows how the open-source agriculture community is growing the next generation of crops. →

GROW KIT + CHILLI PACK

PIMORONI ◈ £39.90 | pimoroni.com

Did the Grow HAT Mini pique your interest? Then you might be interested in a more complete kit, the Grow – Grow Kit + Chilli Pack from Pimoroni. With this kit you get the Grow HAT, three of their moisture sensors, wire, and a pack of three pots, seeds, and soil pellets. Just add a standard Raspberry Pi, or Raspberry Pi Zero, and you're off growing a smart garden with a little kick of three different kinds of hot peppers. And if heat isn't your thing, they also have a herb variety pack!

Right ◈
Learn about electronics and agriculture



VERDICT

Red Hat Co.Lab Farm Kit

Build your own plant care electronics.

8/10

Submersible 3V DC Water Pump

ADAFRUIT ◈ $1.95 | adafruit.com

When you need to move water, and accuracy isn't as important as price, the Submersible 3V DC Water Pump from Adafruit can't be beaten. At under $2, you can easily add this pump to your hydroponics or simple gardening system. In fact, you can add several!

The pump must be submerged in water to operate, and keep in mind that it isn't rated for years of non-stop use. However, it's great to experiment with and to just have some fun since it's so inexpensive. You can control the speed to some extent via PWM and a microcontroller, or you can simply add a basic transistor for easy on/off functions. Looking to create a fountain, or have some simple watering system run for short periods of time? Then check out this incredibly affordable and versatile little water pump.

Left ◈
Get water to your plants

VERDICT

Submersible 3V DC Water Pump

Can't beat the price!

8/10

Chirp!

CATNIP ELECTRONICS ◈ $15 | wemakethings.net

Right ◈
Beep your way to happy houseplants

Forgetting to water your plants, especially the indoor variety, is almost a given for anyone who enjoys a little horticulture. And not everyone wants to electrify their garden by adding Arduinos, Raspberry Pi boards, or the like. Yes, most of our readers do, but not all of them! And that's where the Chirp! plant watering alarm comes into play.

The latest iteration of Chirp! uses some smart programming and capacitive humidity sensing to know when it's time to water your plant. The creator, Catnip electronics, also coats the board to protect it from deteriorating in a wet environment. And even though this isn't as 'smart' as some of the other components in this roundup, Chirp! does know to keep quiet when it's night-time, so it won't become a little annoying bit of electronics when you're trying to sleep. Best of all, Chirp! is open-source, so you can change, hack, or mod it however you'd like. I really like the idea of this simple, open-source, water mentoring system. But I do wish it was solar-powered. It would be my first hack to the Chirp! □

VERDICT

Chirp!

Give your plant a voice.

9/10

Logic analyser

Probe your circuits to find out what's going wrong

By Ben Everard @ben_everard

Logic analysers have a bunch of input pins; they sense the voltage at those pins, and whether it's a 'high' or 'low' voltage, then feed that back to the computer to process the data. Using this, you can then build up a picture of what information is being sent across a digital bus. You can use them to find out what's being sent over I2C, SPI, or other protocols, or simply see how the pin state of your microcontroller changes over time.

They're useful for when you need to be sure exactly what's going on in a microcontroller behind all the layers of abstraction. For example, what exactly is happening on an I2C bus when it's not working? What's going on when you use firmware that does work, but that you don't have the source code to?

These generic logic analysers are available unbranded from a range of direct-from-China websites, usually for around £5. The one we have (like most claim to) can take up to ten million samples at up to 24MHz on up to eight channels.

If you're using Windows, the driver situation is a little odd. You'll need to install Zadig (**zadig.akeo.ie**), and then when you plug in the device, open Zadig and use this to reinstall the driver (go to Options > List all devices, and select the unknown device from the list). Select the WinUSB driver and select Reinstall Driver. At this point, you should be able to open PulseView and connect to the device (using the fx2lafw driver). Sometimes this process didn't work and we had to start again by unplugging the device and restarting Zadig. We're not completely sure why this happens.

The hardware only gets the data into your computer. You need some software to view it

Below
You need to connect the pin headers to your circuit with jumpers

Left
There's no user input on the unit itself

with. Sigrok is a piece of open-source software that handles communication with a whole range of electronics equipment including multimeters, oscilloscopes, and logic analysers such as this one. This includes a command-line interface and a programming interface for writing your own scripts, but for most people, the PulseView graphical interface will be most useful. This provides a visual overview of the data sent in by the logic analyser. The interface is quite spartan, but it provides you with what you need to see, and much of the power of it is hidden away.

By logic analyser standards, our generic machine is a pretty bare-bones device. You can trigger it to start taking readings on a particular event, and you can see the logical values of up to eight channels. That's about it. Although it can take ten million samples, this is a total across all eight channels (there may be a way of disabling capture on some channels, but we couldn't work out how to do it), so really you only get 1.25 million different time points. If you need high-frequency samples, you won't be able to get very much data at all. Bear in mind that you have to sample significantly faster than the digital signal is transmitting, as the logic analyser isn't synchronised with the device. Having a sample rate of at least four times the data rate is advised.

> **Having a sample rate of at least four times** the data rate is advised

This is perhaps the biggest limitation for these devices. If you're working at lower speeds, this may not be too much of a problem. We often find that we have trouble with initialisation more than other bits, so you may find that there's enough space to grab the initialisation data stream, even if you can't capture everything.

On the software side, PulseView really turns this into a useful device. For simple applications, just viewing the traces might be useful enough, but the decoders give you plenty of extra information when working with standard protocols such as I2C and SPI. These annotate your traces with data on what the logical signals mean. There are decoders for the most common (and quite a lot of not-very-common) protocols included by default, though you can write your own decoder if there isn't one already for the protocol you need.

If you're working with low-level devices, sooner or later you will need something to see what's going on at an electrical level. Print statements can only get you so far. Debuggers only get you a little further. When you're working with hardware peripherals such as I2C or Raspberry Pi Pico's PIO, it's not always obvious exactly what's going in or out of a pin. Logic analysers help you really delve into what's happening (though bear in mind that they can only see the logic state as high or low, unlike an oscilloscope which will tell you the actual situation on a channel).

For around £5–£10, these devices offer an amazing boost to your debugging capabilities, but they are still fairly limited. The driver situation on Windows is annoying, but given the price, live-with-able. As long as you're working with slowish signals and can cope with the roughly one million samples available, then these generic logic analysers can really help you get your low-level code working. ◻

DIRECT FROM **SHENZHEN**

REVIEW

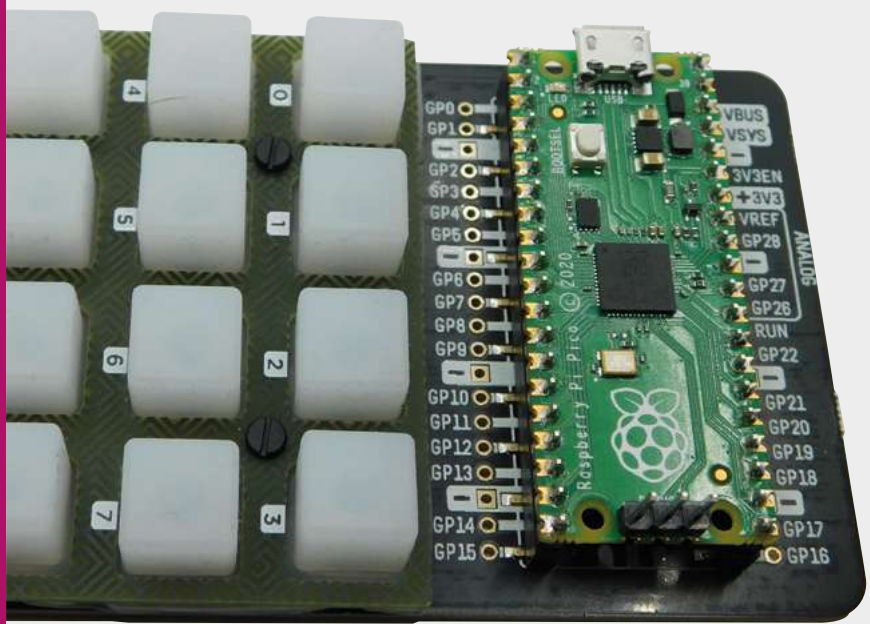
Pico RGB Keypad Base

16 keys and 16 million colours

PIMORONI £21.90 | pimoroni.com

By **Ben Everard** @ben_everard

Adding a button to a microcontroller is fairly straightforward. In fact, it's one of the first things many people do with microcontrollers after blinking an LED. However, adding lots of buttons can be a little tricky. Very quickly you end up with a rat's nest of wiring, and a large proportion of your GPIO gone. That's where Pimoroni's Pico RGB Keypad Base comes in. There are 16 buttons, but these are connected via the I2C port, so the majority of your GPIOs are still free to use. Each key is slightly squishy silicone rubber, so they don't click like keys on a keyboard do, but sort of smush down. The keys are translucent, and under each one is an RGB LED. These are APA102 LEDs driven off the SPI bus, so again, only a few GPIOs are used.



Below
You can connect into any of Pico's outputs when it's in the RGB Keypad Base

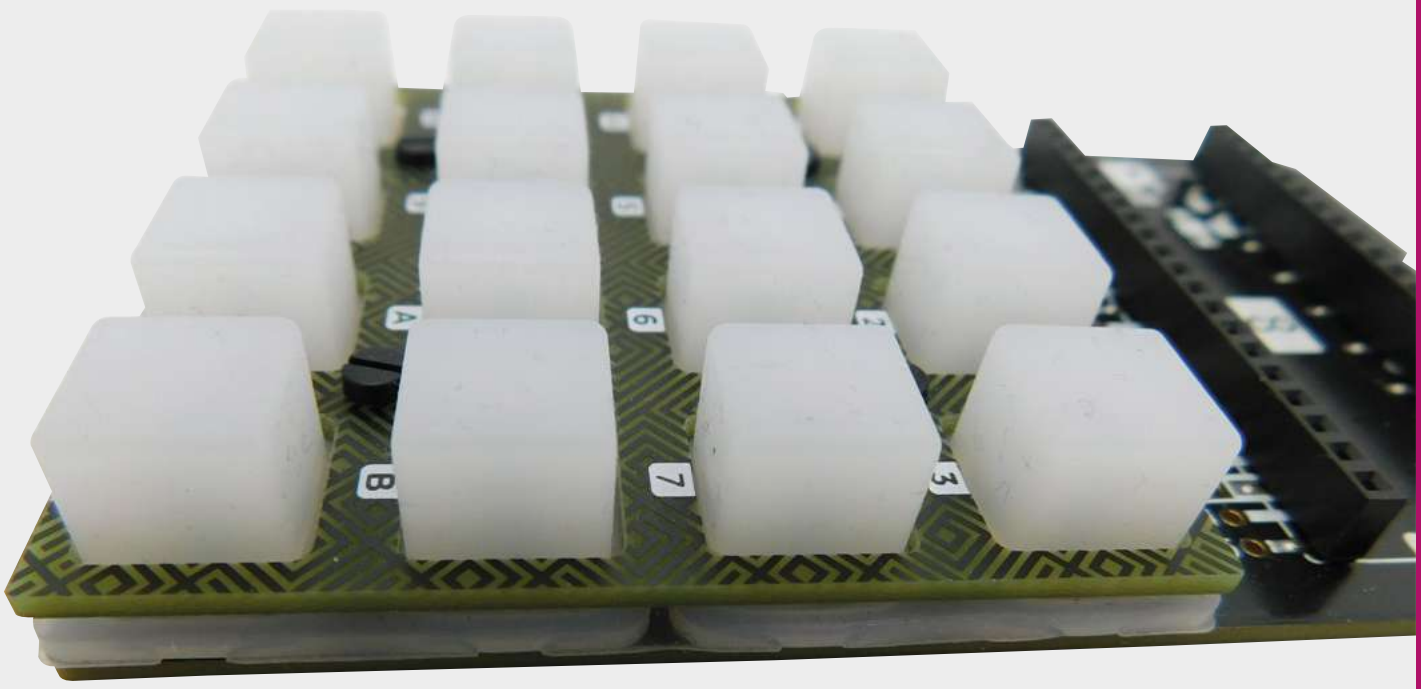
This comes in a few pieces, but assembly is straightforward. The silicone keys go on top of the main PCB, and a second bit of PCB holds the silicone in place. Four bolts stop everything from coming apart. The only thing you have to be careful of is getting the tension on the bolts correct: too tight and they hold the keys pressed; too loose and the keys are prone to getting stuck at an angle.

The Pico RGB Keypad Base breaks out all of the pins, so you can include this expansion in any Pico project with external hardware.

Pimoroni supplies libraries for using this with C++ and MicroPython. Both come with a demo that shows off how to use the hardware, and in both cases it's straightforward. You can set the LEDs and get the button statuses (pressed or not pressed), and that's about it. What you do with these two capabilities is up to you.

If you want to use it with C, you shouldn't have difficulty copying and pasting the relevant parts out of the C++ library into C code. Perhaps though, our favourite way of using the Pico RGB Keypad Base is with CircuitPython. This programming language makes it easy to create USB devices such as keyboards or gamepads. Using this, you can use the Pico RGB Keypad Base to create a games controller, short cuts for your favourite apps, a MIDI controller, or loads of other things.

There's an example, by GitHub user wildestpixel, at **hsmag.cc/RGBKeypad**. In addition, there's more documentation on how to create Human Interface Devices (HIDs) in CircuitPython using this link: **hsmag.cc/HID**.



If you prefer your keys a bit clickier and firmer, you may want to wait for the Keybow 2040 (also from Pimoroni). This board holds 16 mechanical switches back-lit by RGB LEDs and, rather than relying on an external Pico, it has a built-in RP2040, so you get all the power of this new processor without the bulk of a full dev board. While we haven't got our hands on a Keybow 2040 yet, the one downside of this appears to be that the GPIOs of the RP2040 aren't broken out – so if you need access to these, the Pico RGB Keypad Base may be a better choice.

> The only thing you have to be careful of is **getting the tension on the bolts correct**

There's a lot to like about the Pico RGB Keypad Base. If you need a lot of squishy buttons, it's a great choice. The LEDs under each key give it the ability to work as an output as well as an input, so you can show the state of the key if you click it on and off. Or, it could change colour depending on the state of a particular thing, or … well, we'll leave it up to you. There are also a few niggles. It's not quite as easy as we'd like to mix with other hardware. For example, we'd like to be able to use this alongside the Pico Audio add-on, These don't fit together using, for example, the Pico Decker, at least not normally. You can flip the Pico and all the add-ons upside down on the Desker, and they both fit in then, but it's easy to make a mistake which may result in damage to your Pico or one of the addons doing this. A few mounting holes would also make some projects a little tidier.

These are minor complaints, though, and won't impact many projects. If you want colourful lights and 16 squishy buttons, the Pico RGB Keypad is a great addition to Pico, and it makes a whole host of projects easy and cost-effective. □

Above
The squishy silicon keys have a little movement, but not as much as most keyboard keys

Left
The underside of the PCB has places to attach rubber feet

VERDICT

Add keys and colour to your Pico Projects.

9/10

HiFive

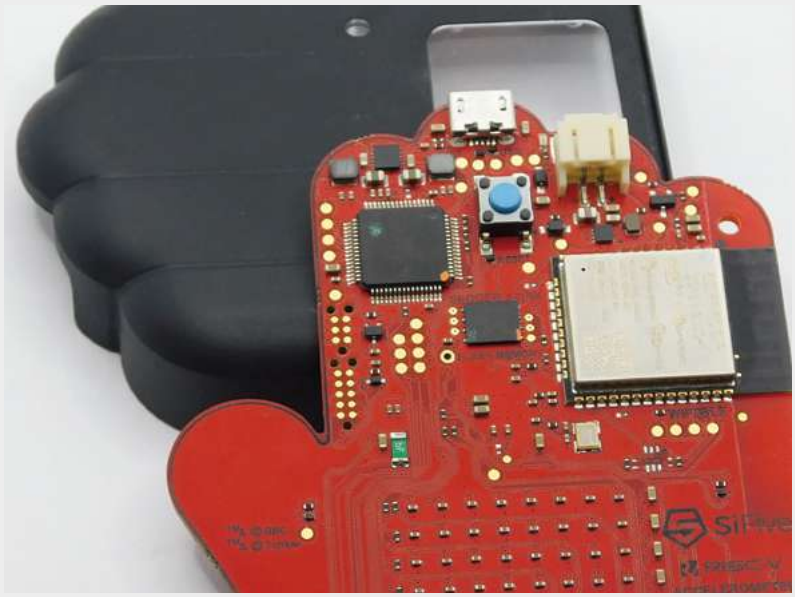
The Doctor will see you now

BBC / TYNKER / SIFIVE FROM £49.50 | hifiveinventor.com

By Ben Everard ben_everard

The HiFive board packs a frankly comical amount of processing onto a microcontroller board. There's a 150MHz, RISC-V-based SiFive FE310 that is the main processor. An ESP32 handles WiFi and Bluetooth, and a SEGGER ARM-based chip handles USB I/O. On top of all that processing power, there's 48 RGB LEDs, a light sensor, an accelerometer, and a compass. If you need more features, there's a micro:bit-style edge connector for hooking up crocodile clips or expanders.

You can get either the bare PCB (for £49.50) or the Inventor Kit (£64.50), which also includes a case, an audio board, crocodile clips, and a battery pack (requires three AA batteries – not included).



Below The ESP32 module gives the device WiFi and Bluetooth connectivity

You can program this using either Scratch-like block coding (following Doctor Who: Planet of Adventure) or MicroPython (following Doctor Who: The Book of Brilliant Inventions). Both of these guides are narrated by Doctor Who actor Jodie Whittaker. There's an additional Intro to MicroPython course, which follows a more traditional style of teaching programming, and Glitch Manor, which is a game to help you build up your programming skills in block coding. You can also create blank projects to do whatever you like.

All this coding is tied into the Tynker programming web interface, which means you have to be on an internet-connected computer to write the code. You can download it to your device straight from the browser (either via USB or Bluetooth). While the internet requirement will annoy some, it means that there's nothing to set up or install – just plug your board in, head to the website, and get started.

This website is behind a login screen, and each HiFive comes with a registration code that lets you set up an account with access to the HiFive resources. Be careful when setting up an account as it doesn't direct you to enter an email address, but without this, there's no way of recovering forgotten passwords. You can create a parent account and link this to one or more child accounts. However, only one child will have access to the guides. Other children (and the parent) will be able to create blank projects for the HiFive Inventor.

The integration of storytelling and programming works well, with the programming sections directly on the web page. For blocks, this is a mostly graphical experience, while with MicroPython the code sections are in line with the text telling the story. The result really is impressive, and the programming feels like a seamless part of the storytelling.

Above The 6×8 RGB matrix is great for displaying text and images

There are ten lessons in each of the guides. It's hard to say how long this will take to go through as there's room for playing and experimentation in each of them. Once you get to the end of the guide, you can create projects and program the board using the Tynker interface. However, some learners may find it a bit of a shock to go from the directed style of the guides to a blank project. At this point, there's less material available for HiFive than for more established programmable boards. While there's certainly the potential for lots of building and experimentation with this board, some learners may feel a little lost once they've finished the guides. That in itself, though, does mean they've worked through one or more programming courses, so getting a learner that far is an achievement in itself.

We liked the guides. They're engaging and have enough storytelling to entertain while also enough programming to be useful. For a young Doctor Who fan, this could be a great way of learning to program. It's especially useful as many children will be able to progress with minimal assistance. The combination of block coding and MicroPython means that quite a wide age range is accommodated. Officially, the kit is for ages seven and up. Obviously this depends on the child and the amount of assistance you can give them, but this feels about right to us.

For people drawn to RISC-V because it's built on the philosophy of open-source hardware, the locked-down interface for coding will no doubt be a disappointment, but they are not the target market for this board (and there are a growing number of more open RISC-V boards catering to their needs).

This board is unashamedly built to help children learn to program. There are a number of other boards that fill this niche, but what makes this one special is the step-by-step Doctor Who-based guide that accompanies it. This is much more of a walkthrough experience than with some other programmable boards aimed at the educational market. There isn't really a right or wrong solution here: it's all about learning styles. Some students will prefer the guided approach of HiFive; some will prefer the more freestyle approach, where you get a programmable board and a range of projects that you can pick and choose.

> "For a young Doctor Who fan, **this could be a great way of learning to program**"

This is expensive when you compare it to other microcontroller boards, but that's a bit of a false comparison. What you're really getting here is a walkthrough of how to code, led by Doctor Who, and it also happens to come with a microcontroller board that you can practise on. For learners who prefer a guided approach, this is one of the best introductions to microcontroller programming we've seen. ◻

VERDICT

A board plus lessons guides young learners through their first steps with microcontrollers.

9/10

next

issue #42 ON SALE 22 APRIL

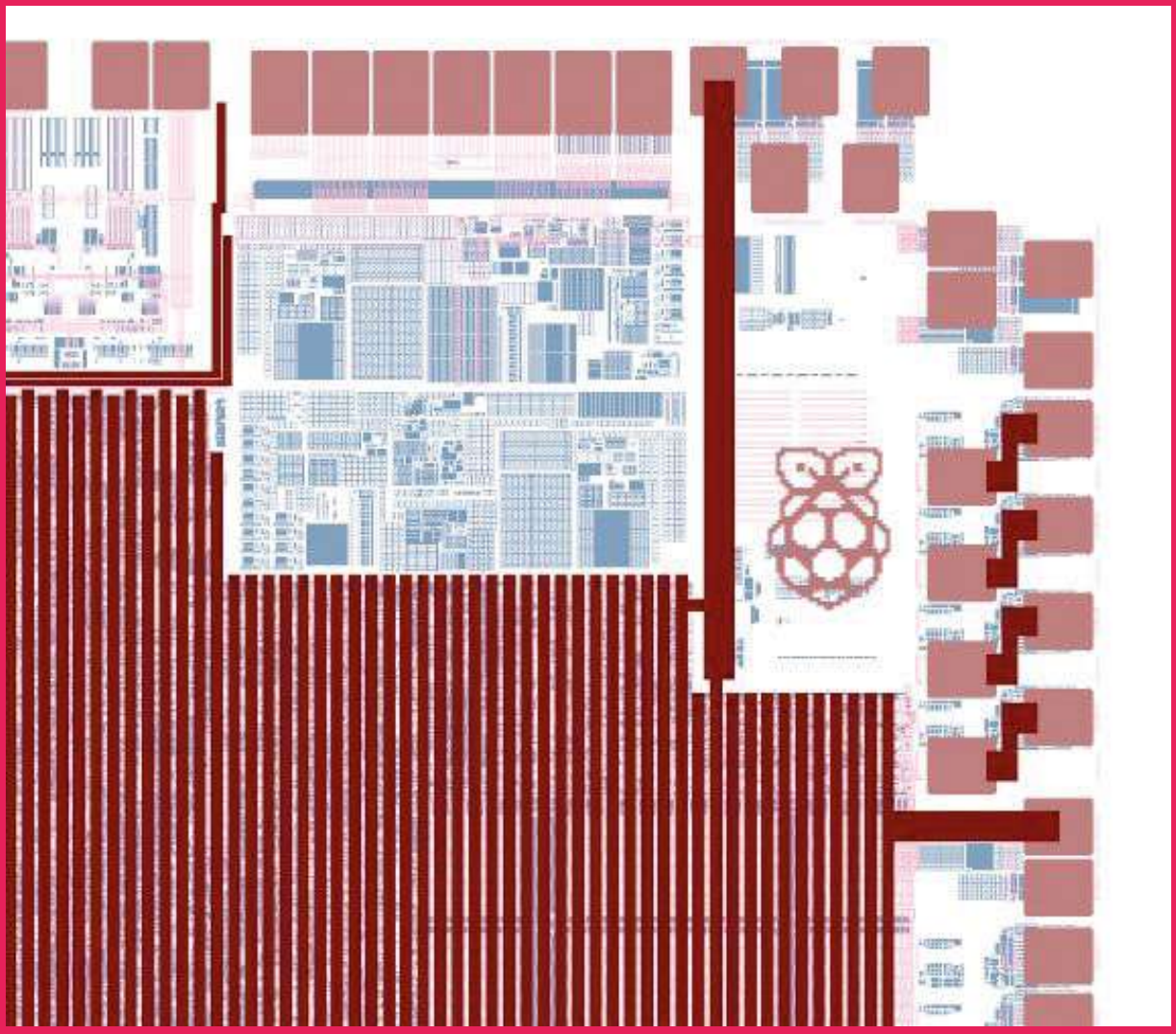
RESTORATION

ALSO

- → RASPBERRY PI PICO
- → 3D PRINTING
- → WOOD-CARVING
- → AND MUCH MORE

DON'T MISS OUT

hsmag.cc/subscribe



There’s a tiny Raspberry Pi logo etched into the silicon powering every Raspberry Pi Pico. This little signature measures just 120 micrometres wide, which is slightly wider than a human hair. There’s a long history of microchip designers putting logos and images on chips – take a look at **zeptobars.com** for some examples from other manufacturers.